allinea

Leaders in parallel software development tools

# Debugging at Scale

## Discipline, Magic, Inspiration and Science

Mark O'Connor

mark@allinea.com

# Schedule Today

| | |
|---|---|
| 09:00 | You, me and a short introduction |
| 10:00 | Debugging by Discipline |
| 10:30 | Debugging by Magic |
| 11:15 | Debugging by Inspiration |
| 12:00 | *Lunch (or zen meditation, I guess)* |
| 13:00 | Debugging by Science |
| 13:45 | Visually-scalable Profiling |
| 14:30 | Q&A |

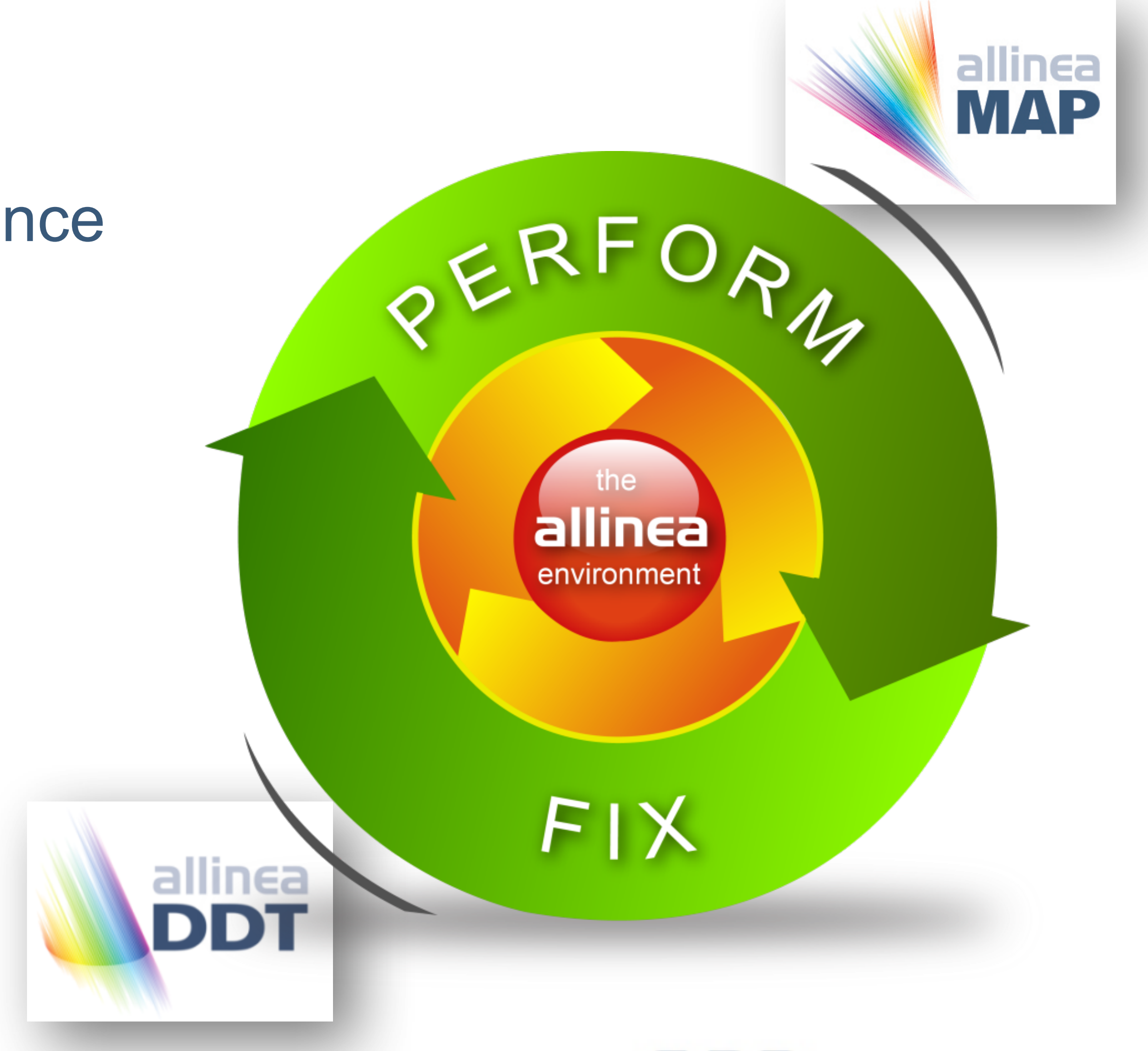
# The Allinea Environment

- A modern integrated environment for HPC developers

- Supporting the lifecycle of application development and improvement
  - **Allinea DDT :** Productively debug code
  - **Allinea MAP :** Enhance application performance

- Designed for productivity
  - Consistent easy to use tools
  - Enables effective HPC development

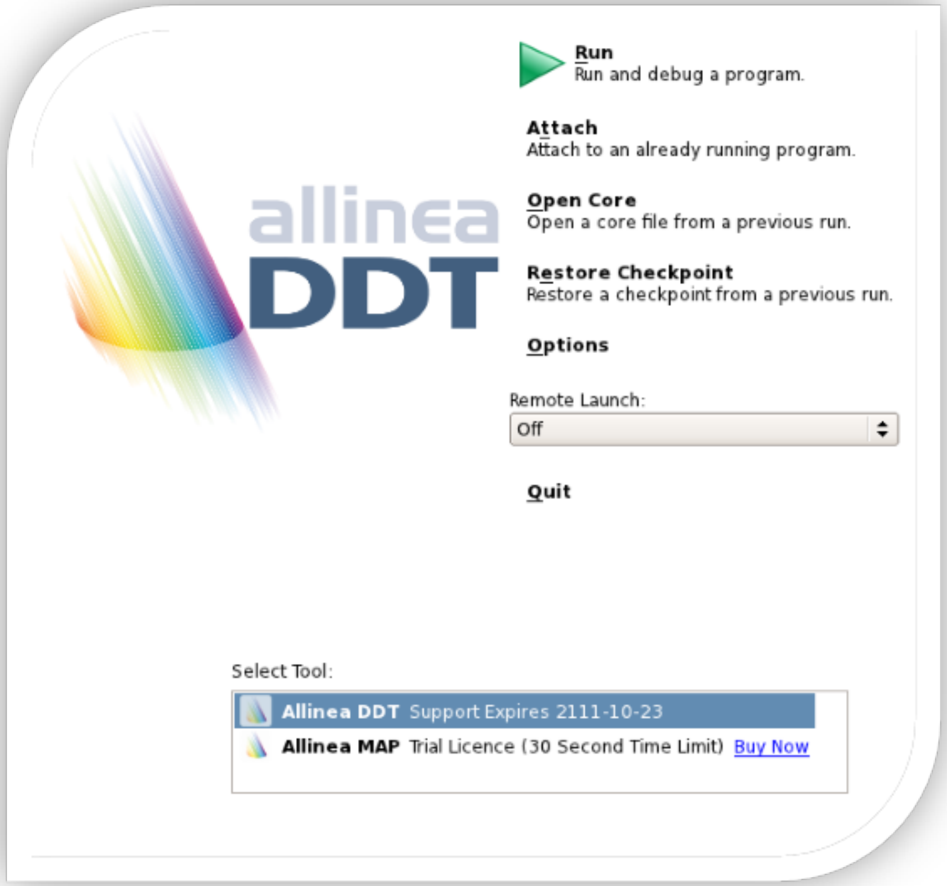
- Improve system usage
  - Fewer failed jobs
  - Higher application performance



www.allinea.com

# Allinea environment
# Fix software problems - fast

- **Graphical debugger designed for:**
  - C/C++, Fortran, MPI, CUDA, SHMEM, UPC
  - Multithreaded code
    - Single address space
  - Multiprocess code
    - Interdependent or independent processes
  - Accelerated codes
    - GPUs, Intel Xeon Phi
  - Any mix of the above



**Run**
Run and debug a program.

**Attach**
Attach to an already running program.

**Open Core**
Open a core file from a previous run.

**Restore Checkpoint**
Restore a checkpoint from a previous run.

**Options**

Remote Launch: Off

**Quit**

Select Tool:

| | |
|---|---|
| **Allinea DDT** | Support Expires 2111-10-23 |
| **Allinea MAP** | Trial Licence (30 Second Time Limit) Buy Now |

- **Slash your time to debug :**
  - Reproduces and triggers your bugs instantly
  - Helps you easily understand where issues come from quickly
  - Helps you to fix them as swiftly as possible

allinea
www.allinea.com

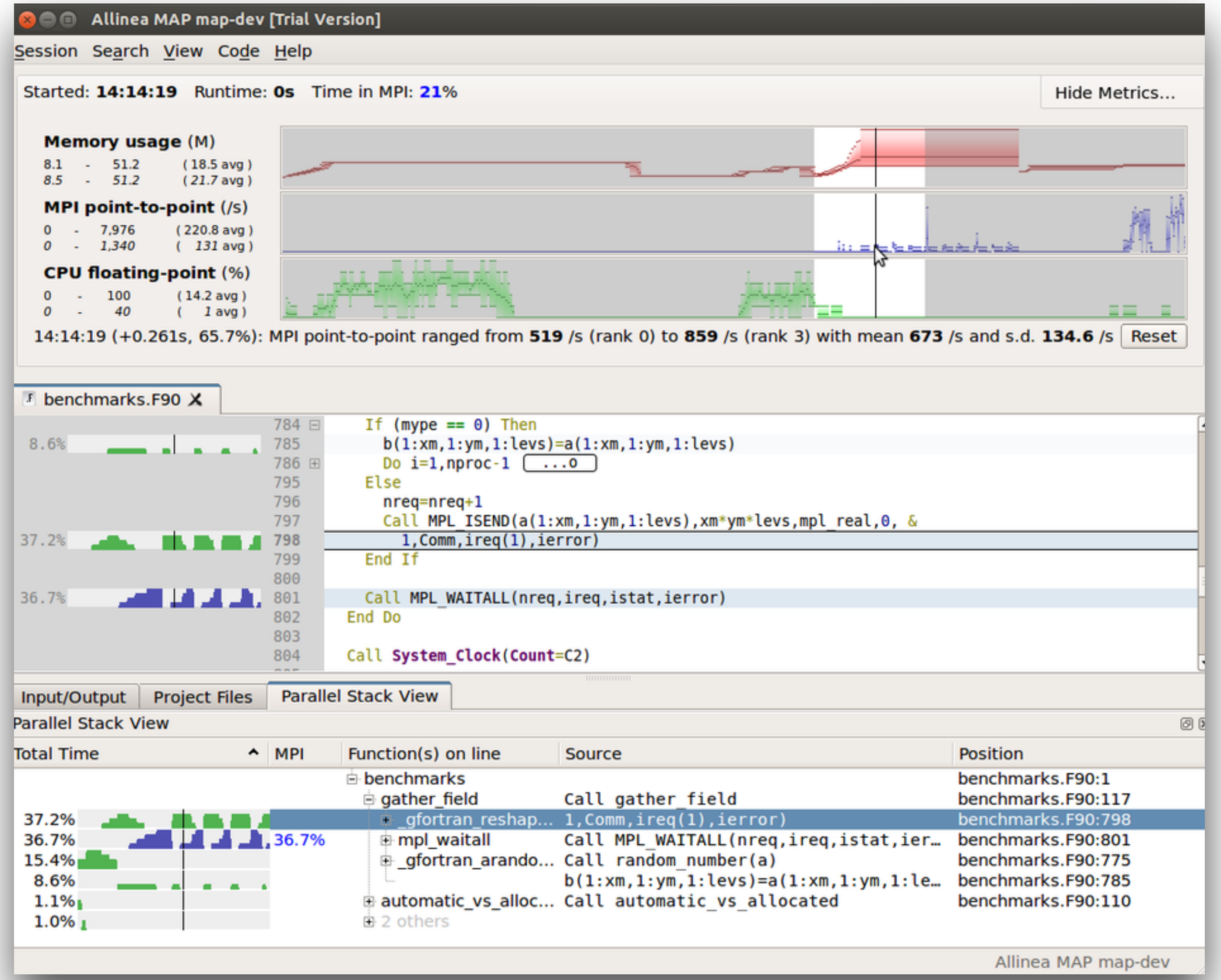
# Allinea MAP: a new kind of profiling

Get a quick, clear performance report

No need to instrument, no need to recompile.

From one process to tens of thousands.

20Mb output files, 5% slowdown.



Started: **14:14:19** Runtime: **0s** Time in MPI: **21**%

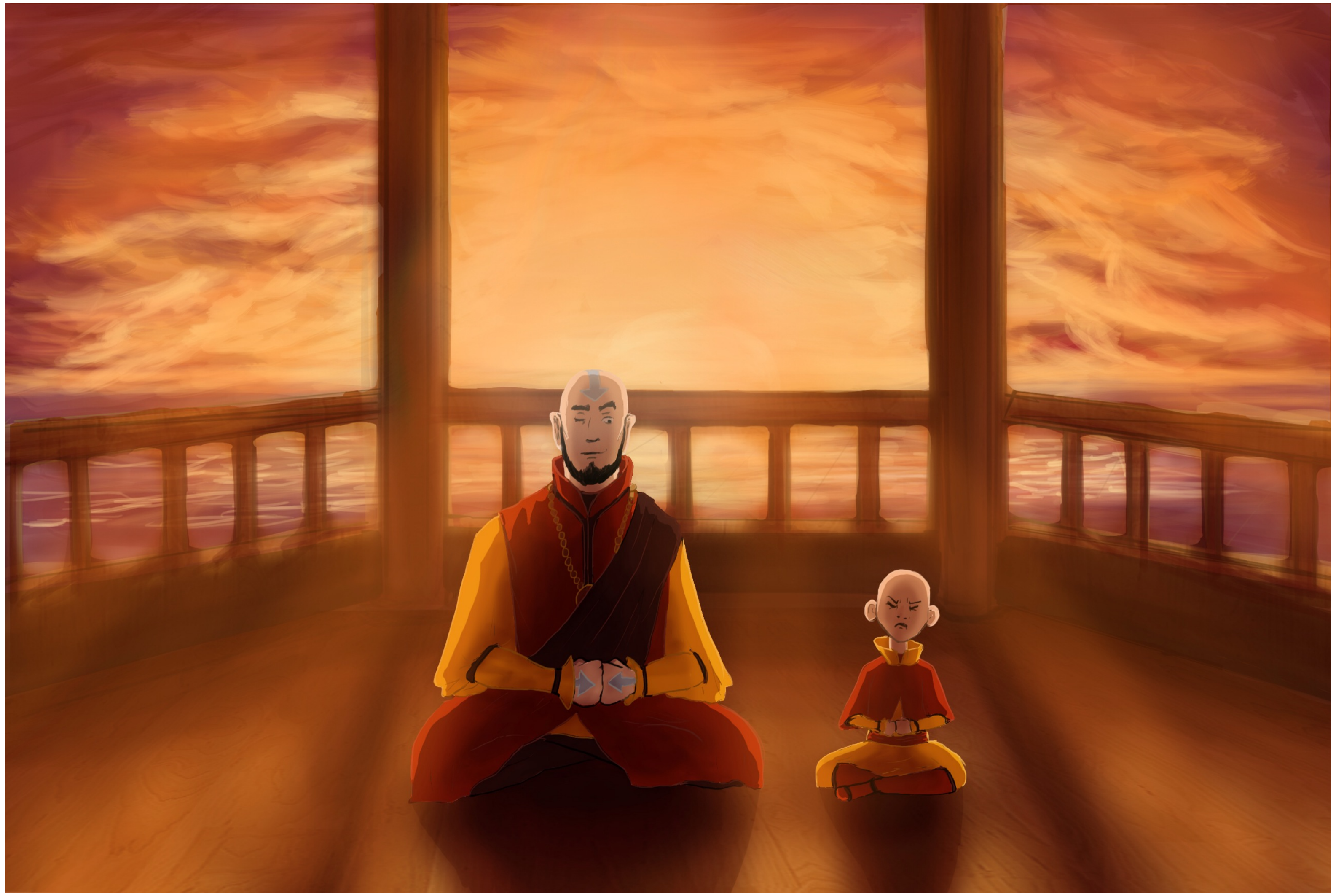
14:14:19 (+0.261s, 65.7%): MPI point-to-point ranged from **519** /s (rank 0) to **859** /s (rank 3) with mean **673** /s and s.d. **134.6** /s

```
    If (mype == 0) Then
      b(1:xm,1:ym,1:levs)=a(1:xm,1:ym,1:levs)
      Do i=1,nproc-1  ...0
    Else
      nreq=nreq+1
      Call MPL_ISEND(a(1:xm,1:ym,1:levs),xm*ym*levs,mpl_real,0, &
        1,Comm,ireq(1),ierror)
    End If

    Call MPL_WAITALL(nreq,ireq,istat,ierror)
  End Do

  Call System_Clock(Count=C2)
```

Parallel Stack View

| Total Time | MPI | Function(s) on line | Source | Position |
|---|---|---|---|---|
| | | benchmarks | | benchmarks.F90:1 |
| | | gather_field | Call gather_field | benchmarks.F90:117 |
| 37.2% | | _gfortran_reshap... | 1,Comm,ireq(1),ierror) | benchmarks.F90:798 |
| 36.7% | 36.7% | mpl_waitall | Call MPL_WAITALL(nreq,ireq,istat,ier... | benchmarks.F90:801 |
| 15.4% | | _gfortran_arando... | Call random_number(a) | benchmarks.F90:775 |
| 8.6% | | | b(1:xm,1:ym,1:levs)=a(1:xm,1:ym,1:le... | benchmarks.F90:785 |
| 1.1% | | automatic_vs_alloc... | Call automatic_vs_allocated | benchmarks.F90:110 |
| 1.0% | | 2 others | | |

# Debugging by Discipline



Three techniques, rigorously applied, will dramatically improve your life.

At least, when it's time to debug.

# Discipline 1: Logbook

```
$ mkdir logs
$ vim logs/segfault-at-4096-procs

When running lu.E.4096 with the trace-4410.dat set,
the job exited with: "An error occurred in MPI_Send
[li346-209:25319] on communicator MPI_COMM_WORLD
MPI_ERR_RANK: invalid rank".

To reproduce: mpiexec -n 4096 lu.W.4096 trace-4410.dat
on supermuc. Seems to happen every time.

* Tried reading core file with gdb, "File truncated"
* Set ulimit -c unlimited and ran again: ...
```

Exactly what happened, including error messages

How to reproduce it

Investigation notes

# Discipline 2: Test script

1. Compile with debug information: -g -O0
2. Submit the job with the correct inputs
3. Wait for it to run...
4. Check the output for errors

So much can - and will - go wrong!
Instead, spend 2 minutes adapting an existing test script that does all of this and prints PASS or FAIL:

```
$ logs/segfault-at-4096-procs.sh
Sep 27 15:29: Queued as job.43214
Sep 27 18:01: Running...
Sep 27 19:29: FAIL
```

# Discipline 3: Source control

```
$ hg init
$ vim .hgignore
    syntax: glob
    *.o
    *~
    *.out
    *.err
$ hg status
$ hg add
$ hg commit -m "Switched to single-precision matrices"
```

You only need to know **four** or **five** commands:

| | |
|---|---|
| `$ hg add main.c` | Add a new file |
| `$ hg commit` | Save the latest changes |
| `$ hg log` | List all previous commits |
| `$ hg update 3` | Update to commit #3 |
| `$ hg revert` | Abandon current changes |

🟩 Source files

🟩 Input files

🟩 Test scripts

🟥 Executables

🟥 Output files

# Summary: Discipline

Efficiency comes through preparation. Debugging a problem is much easier when you can:

| | |
|---|---|
| 1. Track what you've tried so far | **Logbook** |
| 2. Reproduce bugs with a single command | **Test script** |
| 3. Make and undo changes fearlessly | **Source control** |

When these are combined, **magical things** can happen...

# Debugging by Magic

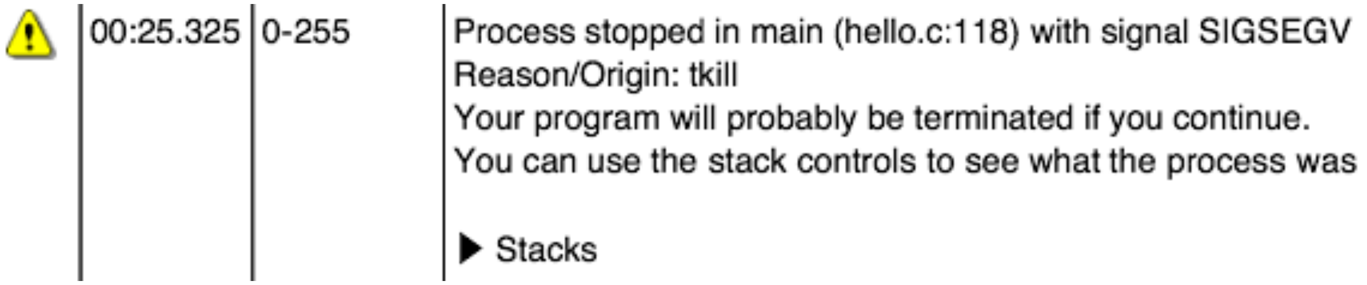
Any technology sufficiently advanced is indistinguishable from magic.

Unpredictable, dangerous, irresistible.

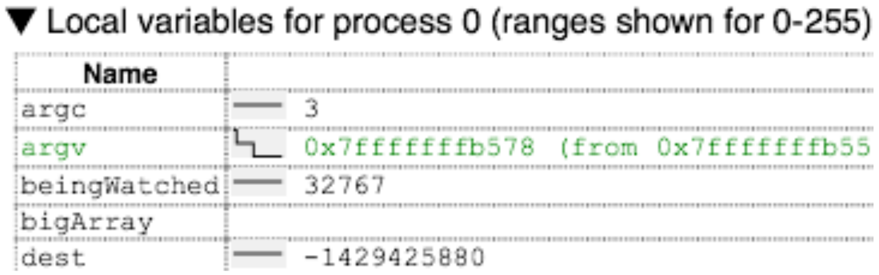
# Favourite Magical Incantations

Program crashes:

```
$ ddt -n 512 -offline log.html myprog.exe arg1 arg2
```



| | | | |
|---|---|---|---|
| ⚠ | 00:25.325 | 0-255 | Process stopped in main (hello.c:118) with signal SIGSEGV<br>Reason/Origin: tkill<br>Your program will probably be terminated if you continue.<br>You can use the stack controls to see what the process was<br><br>▶ Stacks |

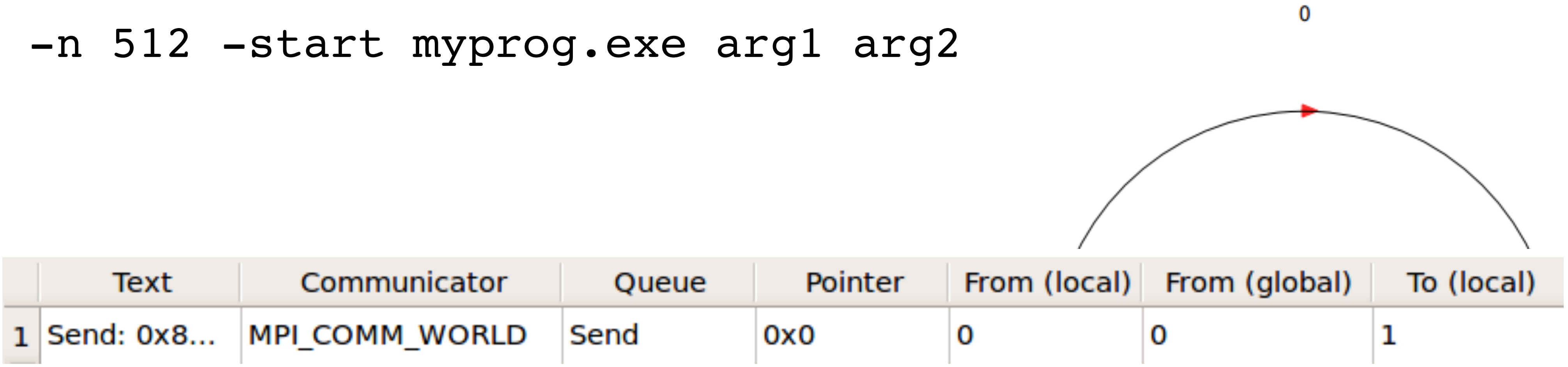


▼ Local variables for process 0 (ranges shown for 0-255)

| Name | |
|---|---|
| argc | 3 |
| argv | 0x7fffffffb578 (from 0x7fffffffb55 |
| beingWatched | 32767 |
| bigArray | |
| dest | -1429425880 |

Program deadlocks:

```
$ ddt -n 512 -start myprog.exe arg1 arg2
```

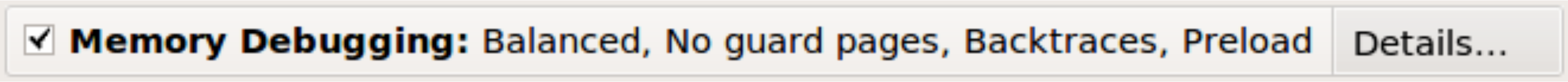


| | Text | Communicator | Queue | Pointer | From (local) | From (global) | To (local) |
|---|---|---|---|---|---|---|---|
| 1 | Send: 0x8... | MPI_COMM_WORLD | Send | 0x0 | 0 | 0 | 1 |

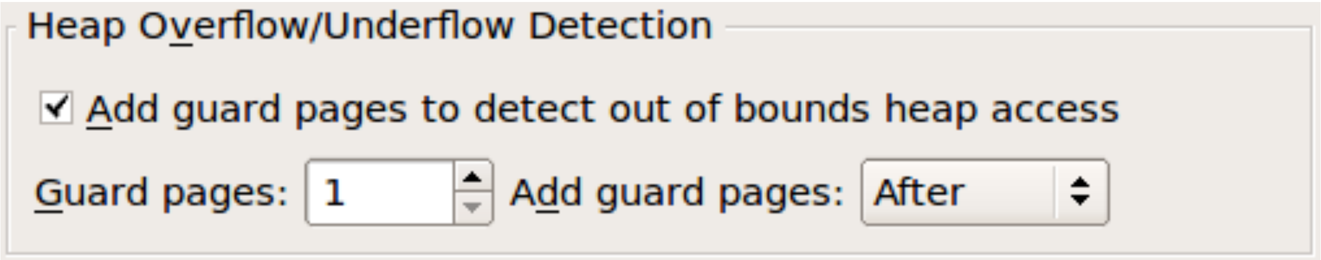
# Favourite Magical Incantations

Suspected memory bugs:

```
$ ddt -n 512 myprog.exe arg1 arg2
```



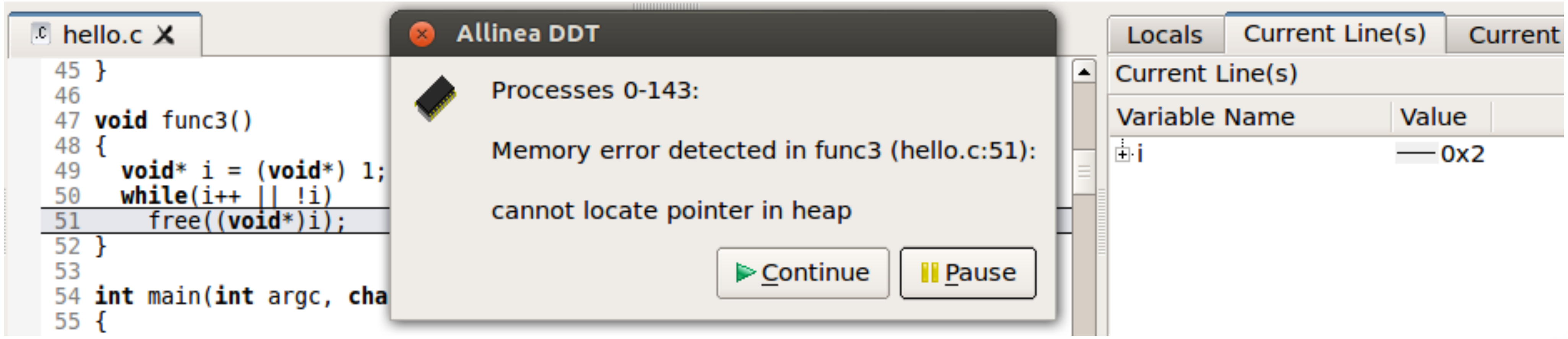
☑ **Memory Debugging:** Balanced, No guard pages, Backtraces, Preload



Heap Overflow/Underflow Detection

☑ Add guard pages to detect out of bounds heap access

Guard pages: 1   Add guard pages: After



```
45 }
46
47 void func3()
48 {
49   void* i = (void*) 1;
50   while(i++ || !i)
51     free((void*)i);
52 }
53
54 int main(int argc, cha
55 {
```

Processes 0-143:

Memory error detected in func3 (hello.c:51):

cannot locate pointer in heap

Current Line(s)

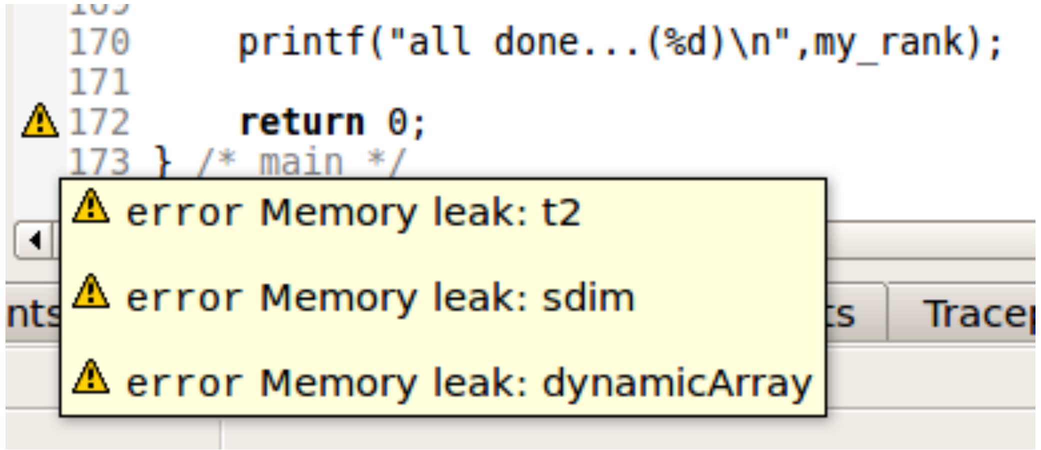
| Variable Name | Value |
|---|---|
| i | 0x2 |

# Protective Magic

Static analysis looks for common mistakes in code:

```
$ /path/to/ddt/libexec/cppcheck
$ /path/to/ddt/libexec/ftnchek          (not a typo!)

  Checking cstartmpi.c...
  [cstartmpi.c:172] (error) Memory leak: t2
```

Also integrated into DDT's GUI and on by default:



```
170     printf("all done...(%d)\n",my_rank);
171
172     return 0;
173 } /* main */
```

⚠ error Memory leak: t2

⚠ error Memory leak: sdim

⚠ error Memory leak: dynamicArray

# Disciplined Magic

A magical servant!

```
$ hg bisect --bad
```

"The current version has a problem"

```
$ hg bisect --good 4
```

"It worked back in version 4"

```
$ hg bisect -c logs/my-test.sh
```

Master, you broke it when you committed: "Switched to single-precision matrices"

# Summary: Magic

A good first step - can be a quick and easy way to fix:

Crashes

Deadlock

Memory problems

Use source control and tests to create a magical servant:

```
$ hg bisect --bad
$ hg bisect --good 4
$ hg bisect -c logs/my-test.sh
$ hg log -pr <changeset id>
```

**Bonus** - static analysis:
cppcheck / ftncheck

# Debugging by Inspiration

Look at the problem,
see the solution.

Vertrauen ist gut,
trust your instincts.

Kontrolle ist besser,
test if they're right

# Inspiration: Test your instincts

> "Ah, I bet that's because rl_x is negative, so then the y offset will be wrong, and ..."

Check to see if you're right before changing the code:

```
$ ddt -n 64 -offline log.html -trace-at blas.c:412,rl_x myprog.exe
```

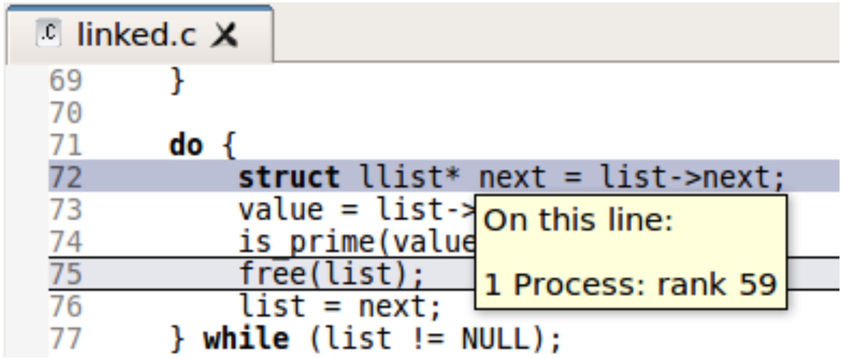
If your instinct is right, great! If it's wrong, **log it**:

```
$ cat >> logs/incorrect-y-merge
   Hypothesis: Negative rl_x in blas.c:412 corrupts the y offset
   Observation: according to -trace-at, rl_x = 4.113 in blas.c:412
```

# Inspiration: Just look at the problem

Explore the data and program flow in an **interactive** session:

```
$ ddt -n 192 -start myprog.exe arg1 arg2
```

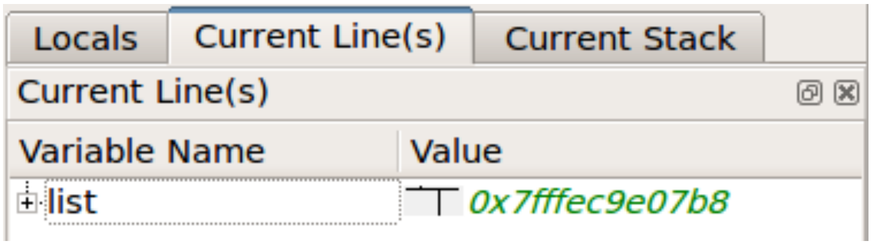


linked.c

```
    }

    do {
        struct llist* next = list->next;
        value = list->
        is_prime(value
        free(list);
        list = next;
    } while (list != NULL);
```

On this line:

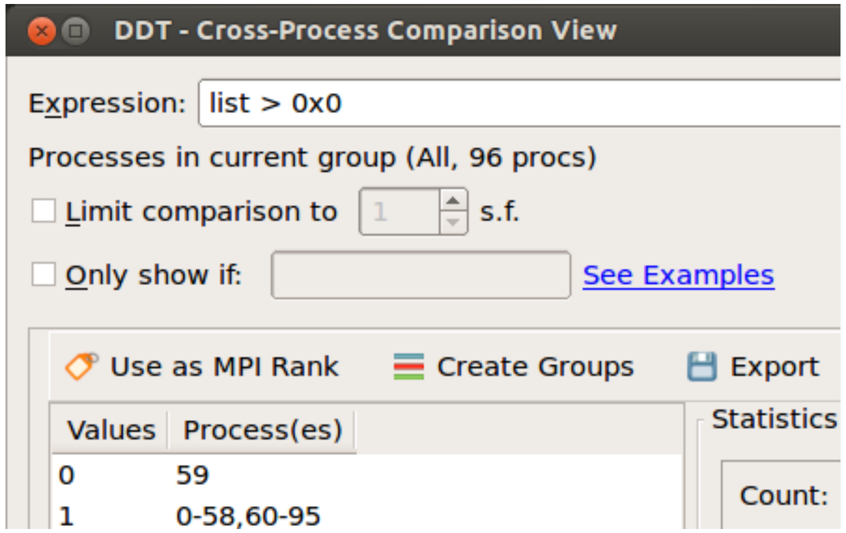
1 Process: rank 59



Locals | Current Line(s) | Current Stack

Current Line(s)

| Variable Name | Value |
|---|---|
| list | 0x7fffec9e07b8 |



DDT - Cross-Process Comparison View

Expression: list > 0x0

Processes in current group (All, 96 procs)

□ Limit comparison to 1 s.f.

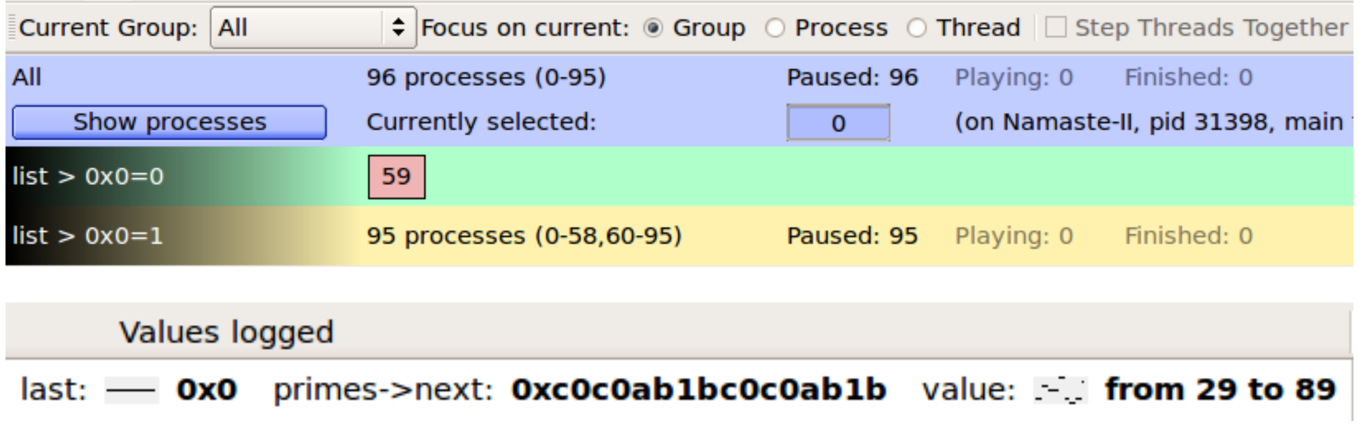
□ Only show if: See Examples

Use as MPI Rank | Create Groups | Export

| Values | Process(es) |
|---|---|
| 0 | 59 |
| 1 | 0-58,60-95 |

Statistics

Count:

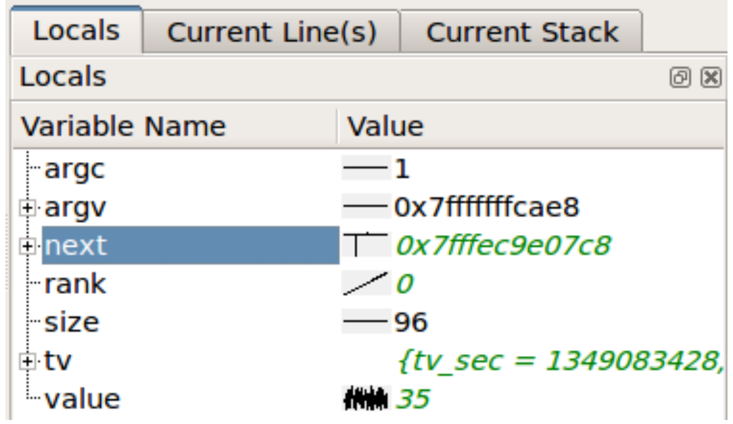


Current Group: All | Focus on current: ● Group ○ Process ○ Thread □ Step Threads Together

| | | | | |
|---|---|---|---|---|
| All | 96 processes (0-95) | Paused: 96 | Playing: 0 | Finished: 0 |
| Show processes | Currently selected: 0 | (on Namaste-II, pid 31398, main | | |
| list > 0x0=0 | 59 | | | |
| list > 0x0=1 | 95 processes (0-58,60-95) | Paused: 95 | Playing: 0 | Finished: 0 |

Values logged

last: 0x0 primes->next: 0xc0c0ab1bc0c0ab1b value: from 29 to 89

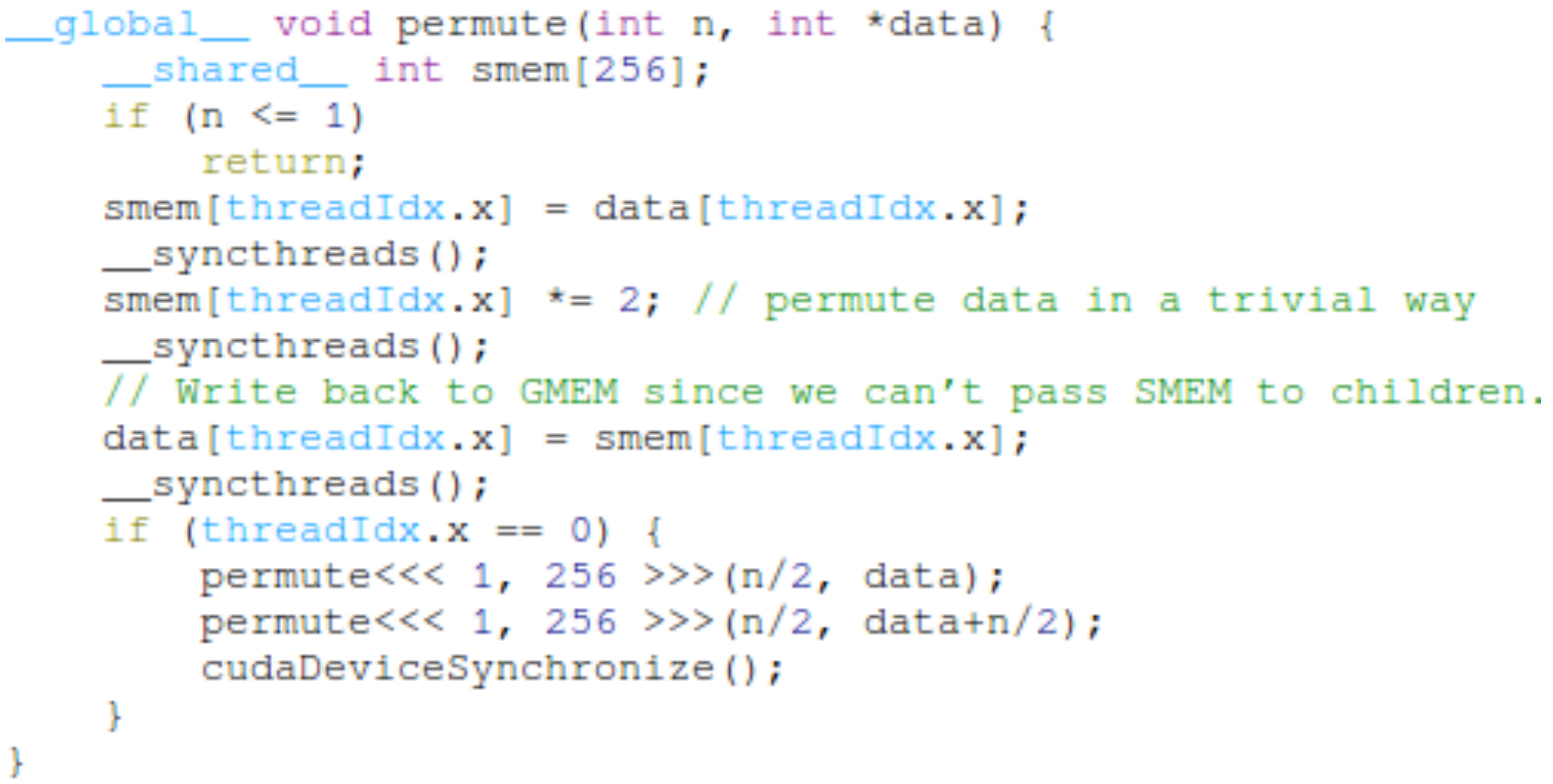


Locals | Current Line(s) | Current Stack

Locals

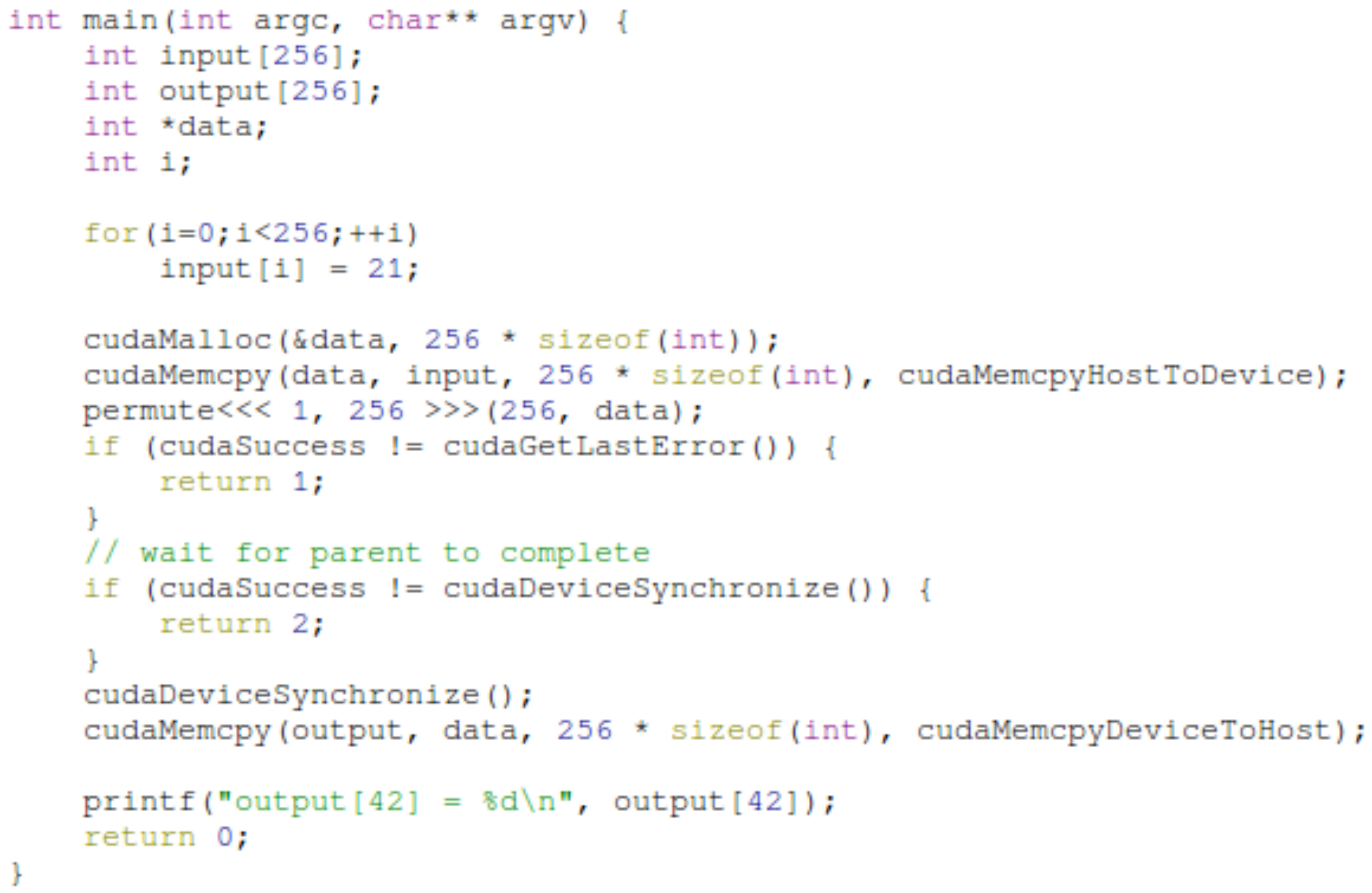
| Variable Name | Value |
|---|---|
| argc | 1 |
| argv | 0x7fffffffcae8 |
| next | 0x7fffec9e07c8 |
| rank | 0 |
| size | 96 |
| tv | {tv_sec = 1349083428, |
| value | 35 |

# Example: CUDA Dynamic Parallelism



```
__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}
```

# Debugging Dynamic Parallelism

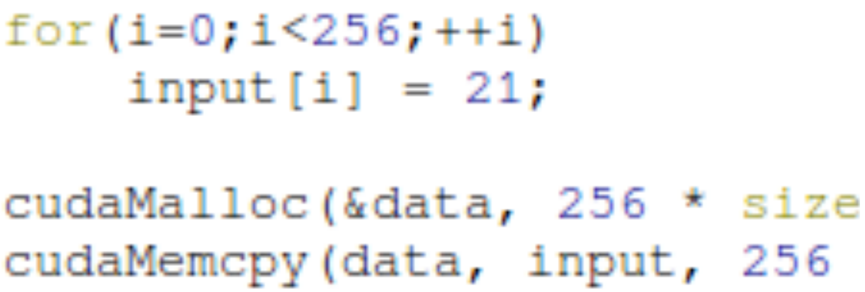


```
int main(int argc, char** argv) {
    int input[256];
    int output[256];
    int *data;
    int i;

    for(i=0;i<256;++i)
        input[i] = 21;

    cudaMalloc(&data, 256 * sizeof(int));
    cudaMemcpy(data, input, 256 * sizeof(int), cudaMemcpyHostToDevice);
    permute<<< 1, 256 >>>(256, data);
    if (cudaSuccess != cudaGetLastError()) {
        return 1;
    }
    // wait for parent to complete
    if (cudaSuccess != cudaDeviceSynchronize()) {
        return 2;
    }
    cudaDeviceSynchronize();
    cudaMemcpy(output, data, 256 * sizeof(int), cudaMemcpyDeviceToHost);

    printf("output[42] = %d\n", output[42]);
    return 0;
}
```

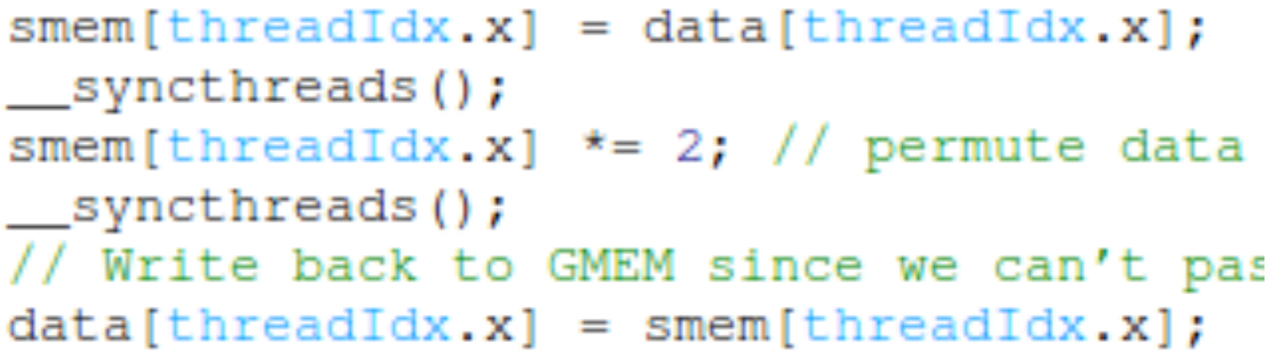
# Debugging Dynamic Parallelism

```
cuda/dynpar $ nvcc -g -G -O0 -arch=sm_35 -rdc=true ./permute
.cu -o permute -lcudadevrt
cuda/dynpar $ ./permute
output[42] = 0
```

wait, what?

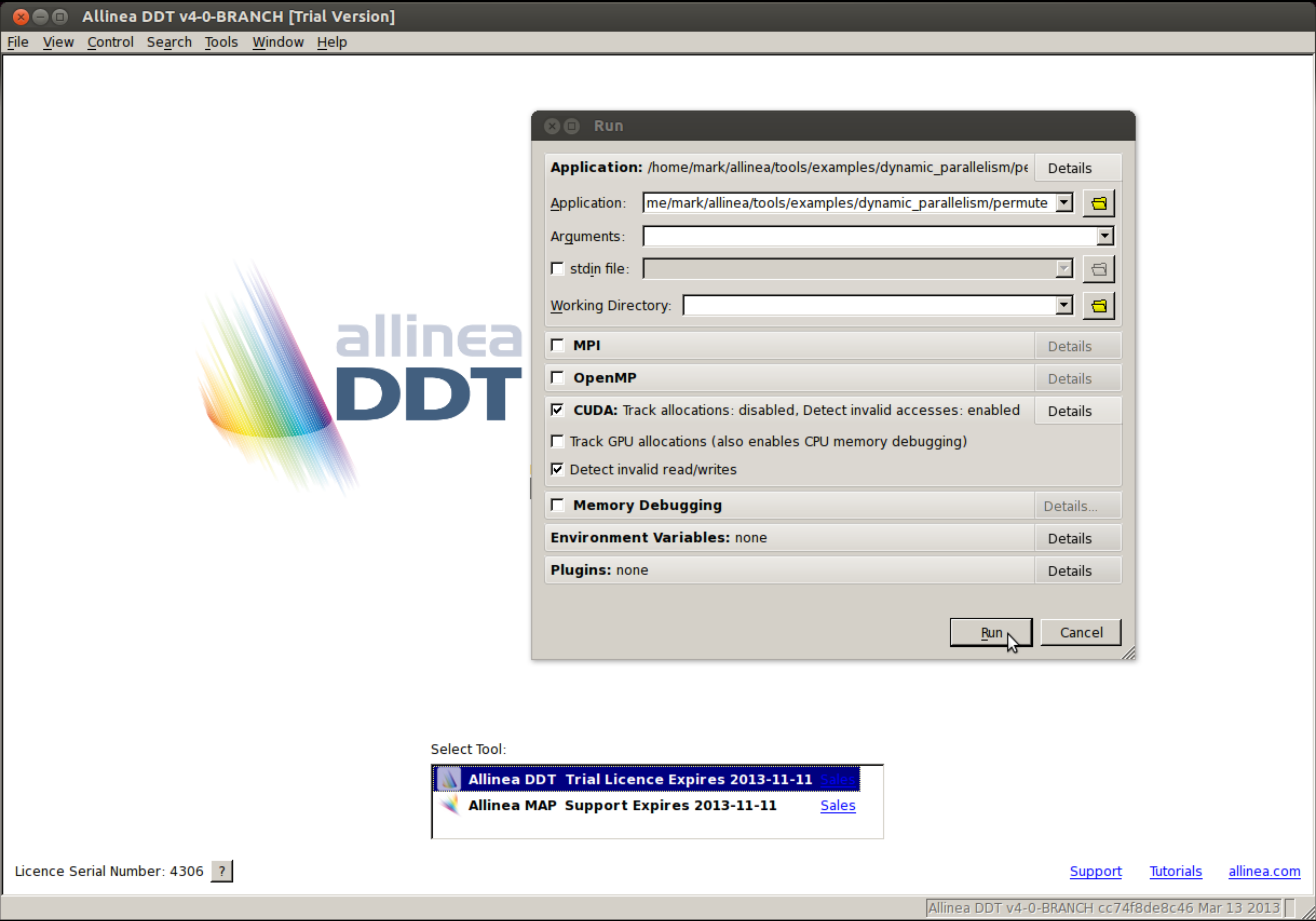


```
for(i=0;i<256;++i)
    input[i] = 21;

cudaMalloc(&data, 256 * size
cudaMemcpy(data, input, 256
```



```
smem[threadIdx.x] = data[threadIdx.x];
__syncthreads();
smem[threadIdx.x] *= 2; // permute data
__syncthreads();
// Write back to GMEM since we can't pas
data[threadIdx.x] = smem[threadIdx.x];
```

```
cuda/dynpar $ ddt ./permute
```



Allinea DDT v4-0-BRANCH [Trial Version]

Run

**Application:** /home/mark/allinea/tools/examples/dynamic_parallelism/pe

Application: me/mark/allinea/tools/examples/dynamic_parallelism/permute

Arguments:

□ stdin file:

Working Directory:

□ **MPI**

□ **OpenMP**

☑ **CUDA:** Track allocations: disabled, Detect invalid accesses: enabled

□ Track GPU allocations (also enables CPU memory debugging)

☑ Detect invalid read/writes

□ **Memory Debugging**

**Environment Variables:** none

**Plugins:** none

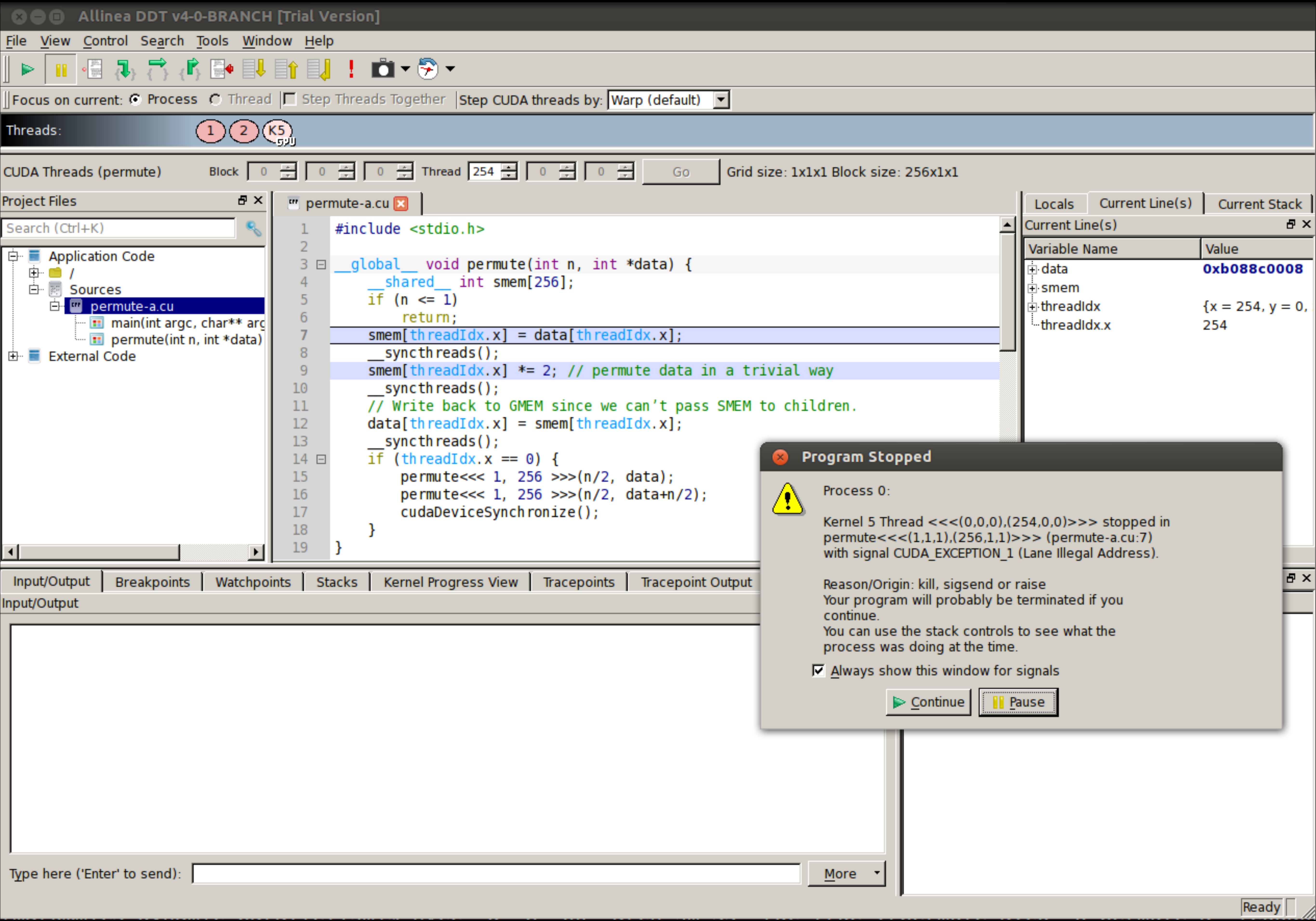
Select Tool:

**Allinea DDT Trial Licence Expires 2013-11-11** Sales

**Allinea MAP Support Expires 2013-11-11** Sales

Licence Serial Number: 4306

Allinea DDT v4-0-BRANCH cc74f8de8c46 Mar 13 2013


Allinea DDT v4-0-BRANCH [Trial Version]
File View Control Search Tools Window Help
Threads: 1 2 K5 GPU
Project Files
Search (Ctrl+K)
Application Code
/
Sources
permute-a.cu
main(int argc, char** arg
permute(int n, int *data)
External Code
permute-a.cu
Locals Current Line(s) Current Stack
Input/Output Breakpoints Watchpoints Stacks Kernel Progress View Tracepoints Tracepoint Output
Type here ('Enter' to send):
More
Ready

Focus on current: ● Process ○ Thread ☐ Step Threads Together

Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 254 0 0 Go Grid size: 1x1x1 Block size: 256x1x1

```
#include <stdio.h>

__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}
```

Current Line(s)

| Variable Name | Value |
|---|---|
| data | 0xb088c0008 |
| smem | |
| threadIdx | {x = 254, y = 0, |
| threadIdx.x | 254 |

Program Stopped

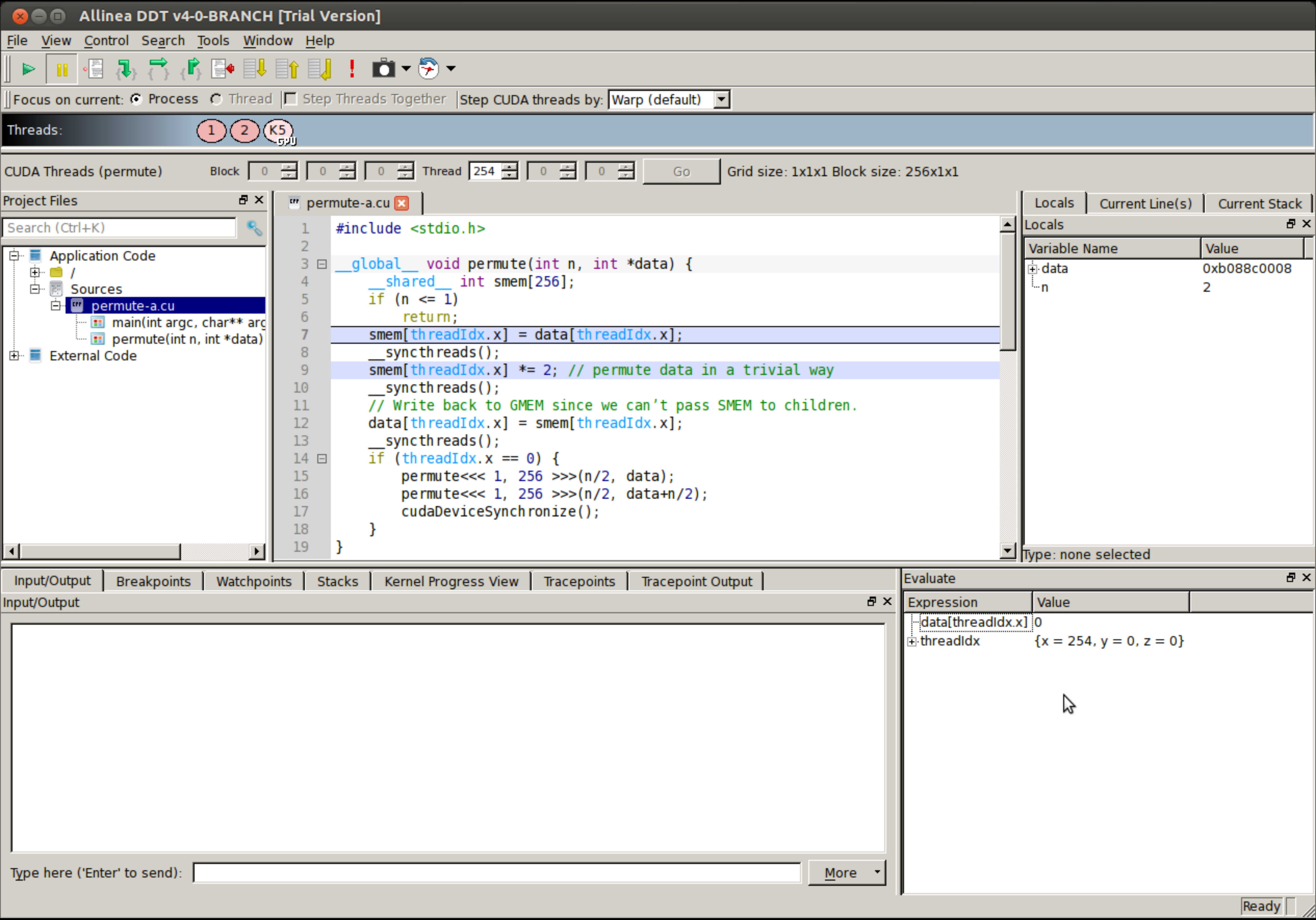
Process 0:

Kernel 5 Thread <<<(0,0,0),(254,0,0)>>> stopped in permute<<<(1,1,1),(256,1,1)>>> (permute-a.cu:7) with signal CUDA_EXCEPTION_1 (Lane Illegal Address).

Reason/Origin: kill, sigsend or raise
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.

☑ Always show this window for signals

Continue Pause



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 254 0 0 Grid size: 1x1x1 Block size: 256x1x1

```
#include <stdio.h>

__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}
```

Locals

| Variable Name | Value |
|---|---|
| data | 0xb088c0008 |
| n | 2 |

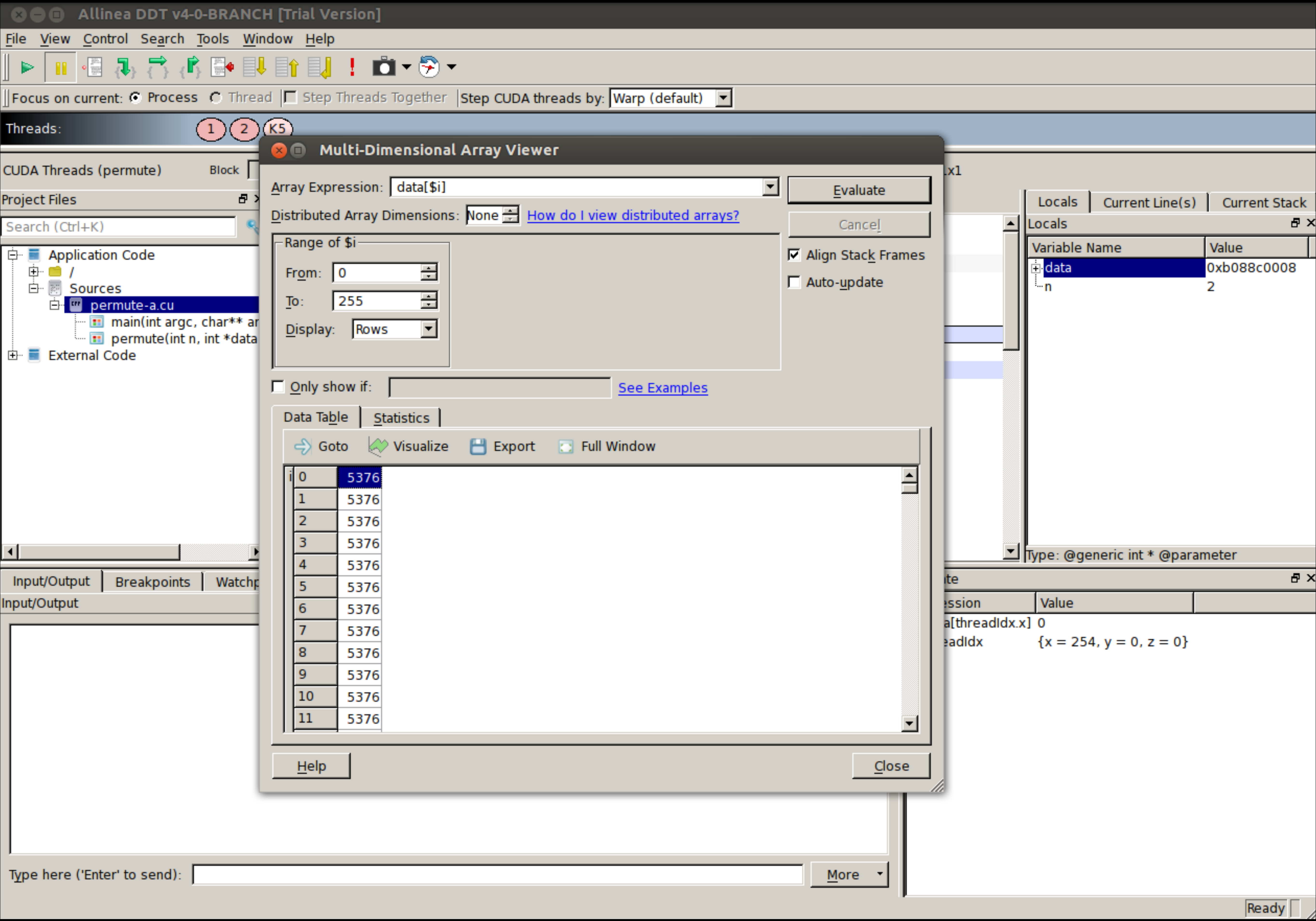
Type: none selected

Input/Output

Type here ('Enter' to send):

Evaluate

| Expression | Value |
|---|---|
| data[threadIdx.x] | 0 |
| threadIdx | {x = 254, y = 0, z = 0} |



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block

Search (Ctrl+K)

- Application Code
  - /
  - Sources
    - permute-a.cu
      - main(int argc, char** ar
      - permute(int n, int *data
- External Code

Multi-Dimensional Array Viewer

Array Expression: data[$i]

Distributed Array Dimensions: None How do I view distributed arrays?

Range of $i

From: 0

To: 255

Display: Rows

☑ Align Stack Frames

□ Auto-update

□ Only show if: See Examples

| i | |
|---|---|
| 0 | 5376 |
| 1 | 5376 |
| 2 | 5376 |
| 3 | 5376 |
| 4 | 5376 |
| 5 | 5376 |
| 6 | 5376 |
| 7 | 5376 |
| 8 | 5376 |
| 9 | 5376 |
| 10 | 5376 |
| 11 | 5376 |

Locals

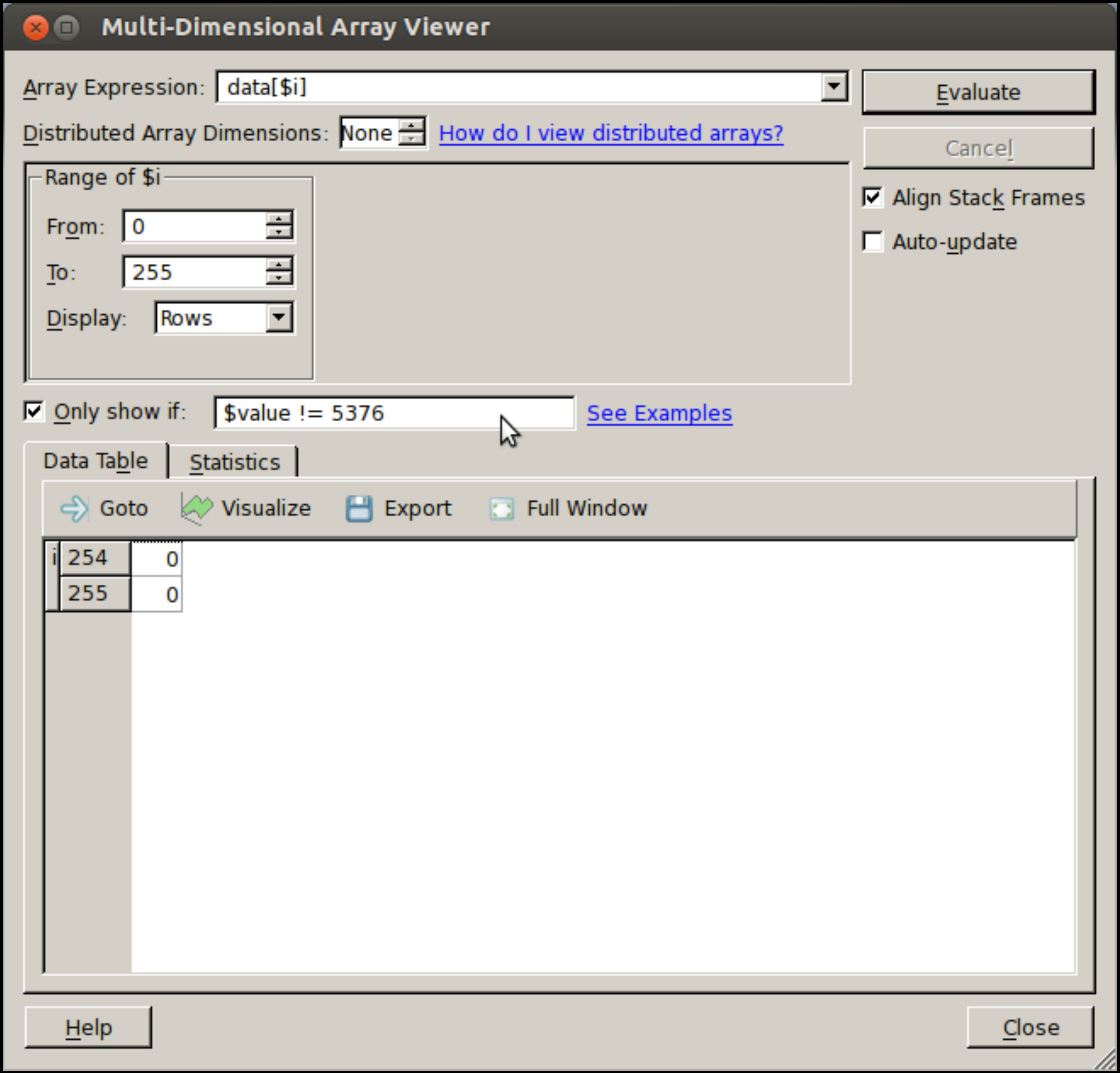
| Variable Name | Value |
|---|---|
| data | 0xb088c0008 |
| n | 2 |

Type: @generic int * @parameter

| ession | Value |
|---|---|
| a[threadIdx.x] | 0 |
| eadIdx | {x = 254, y = 0, z = 0} |

Input/Output

Type here ('Enter' to send):



Multi-Dimensional Array Viewer

Array Expression: data[$i]

Distributed Array Dimensions: None

How do I view distributed arrays?

Range of $i

From: 0

To: 255

Display: Rows

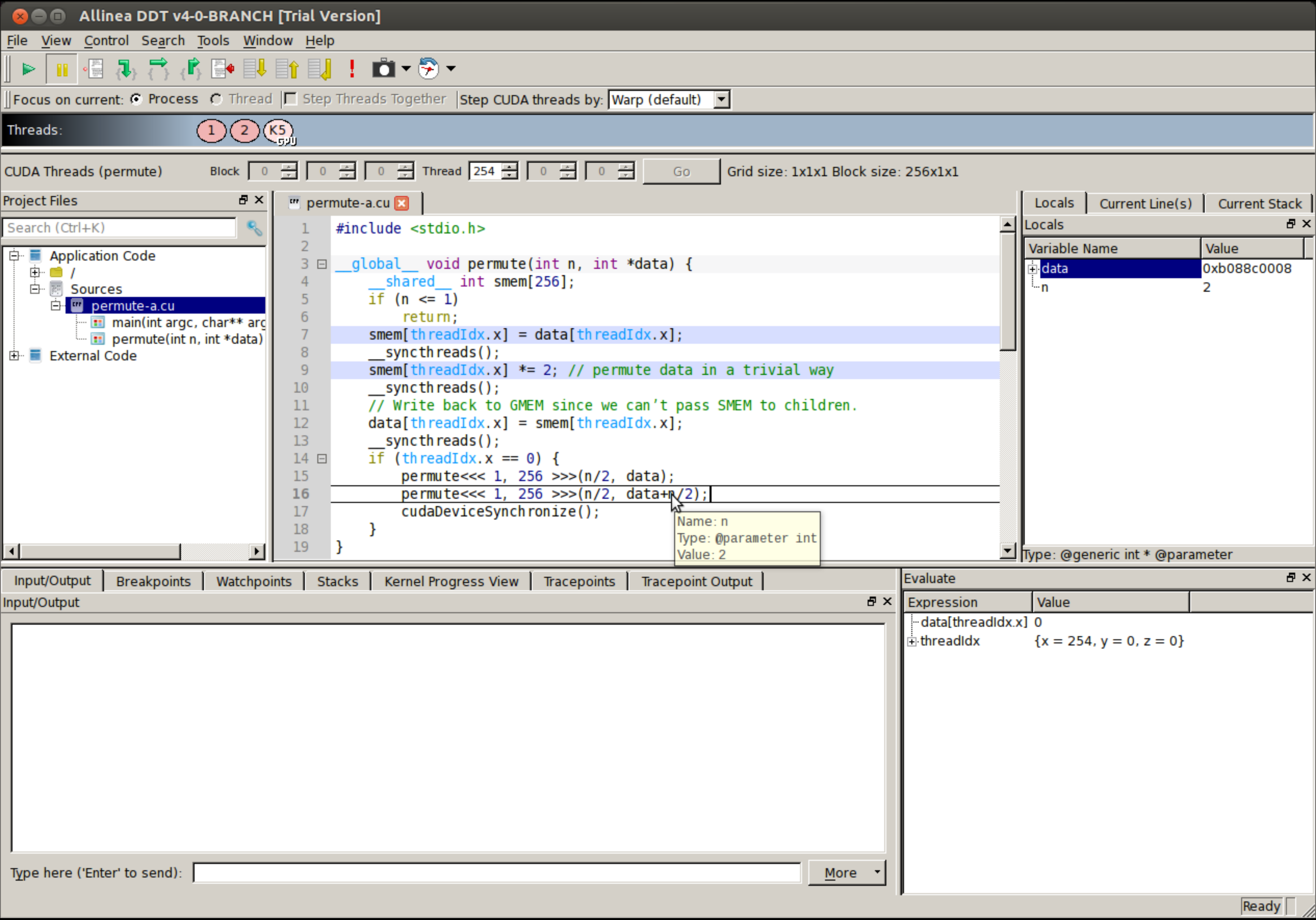
☑ Align Stack Frames

☐ Auto-update

☑ Only show if: $value != 5376

See Examples

| i | |
|---|---|
| 254 | 0 |
| 255 | 0 |


Allinea DDT v4-0-BRANCH [Trial Version]
File View Control Search Tools Window Help
Threads: 1 2 K5 GPU
Project Files
Search (Ctrl+K)
Application Code
/
Sources
permute-a.cu
main(int argc, char** arc
permute(int n, int *data)
External Code
permute-a.cu
Locals Current Line(s) Current Stack
Input/Output Breakpoints Watchpoints Stacks Kernel Progress View Tracepoints Tracepoint Output
Input/Output
Type here ('Enter' to send):
More
Evaluate
Ready

Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 254 0 0 Go Grid size: 1x1x1 Block size: 256x1x1

```
#include <stdio.h>

__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}
```

Name: n
Type: @parameter int
Value: 2

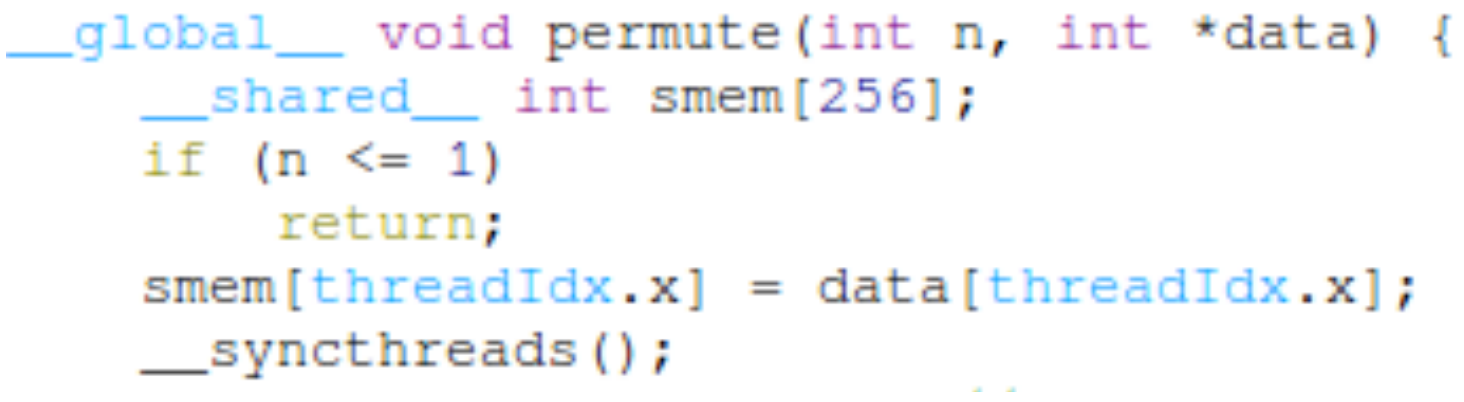
Locals

| Variable Name | Value |
|---|---|
| data | 0xb088c0008 |
| n | 2 |

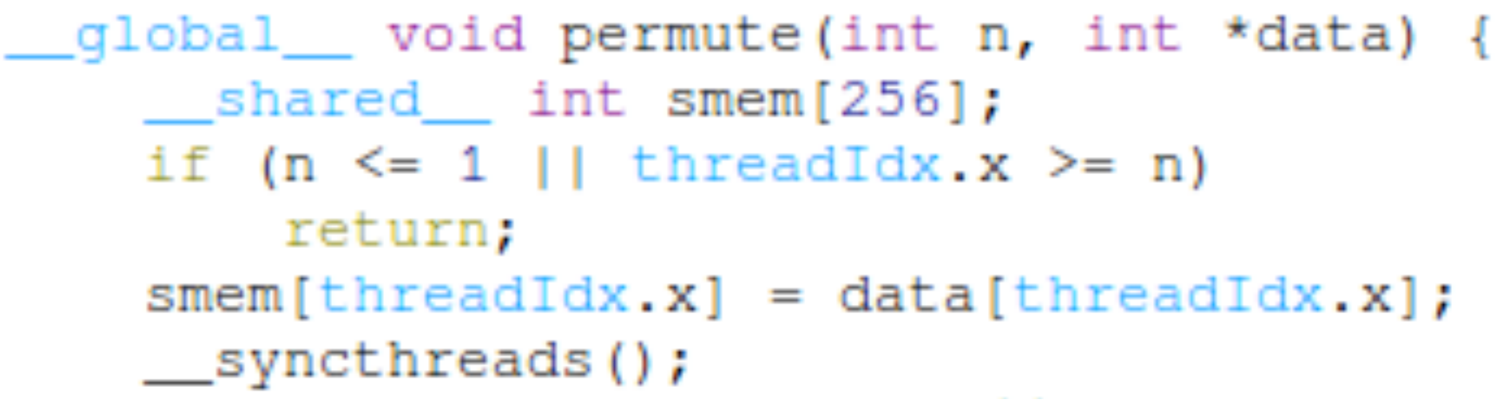
Type: @generic int * @parameter

| Expression | Value |
|---|---|
| data[threadIdx.x] | 0 |
| threadIdx | {x = 254, y = 0, z = 0} |

# Debugging Dynamic Parallelism

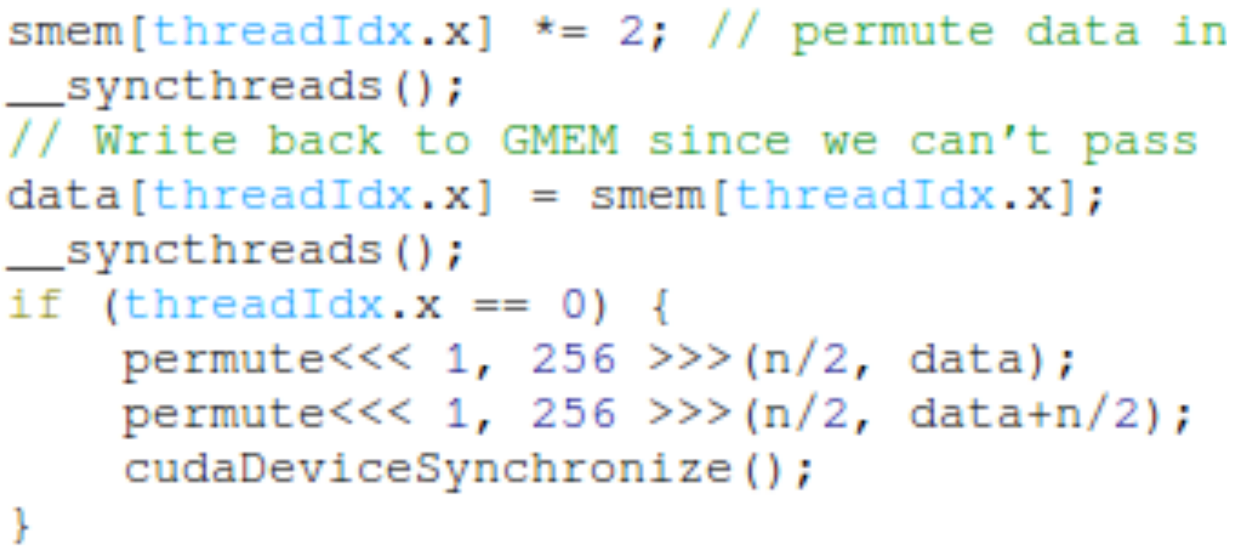


```
__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
```



```
__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1 || threadIdx.x >= n)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
```

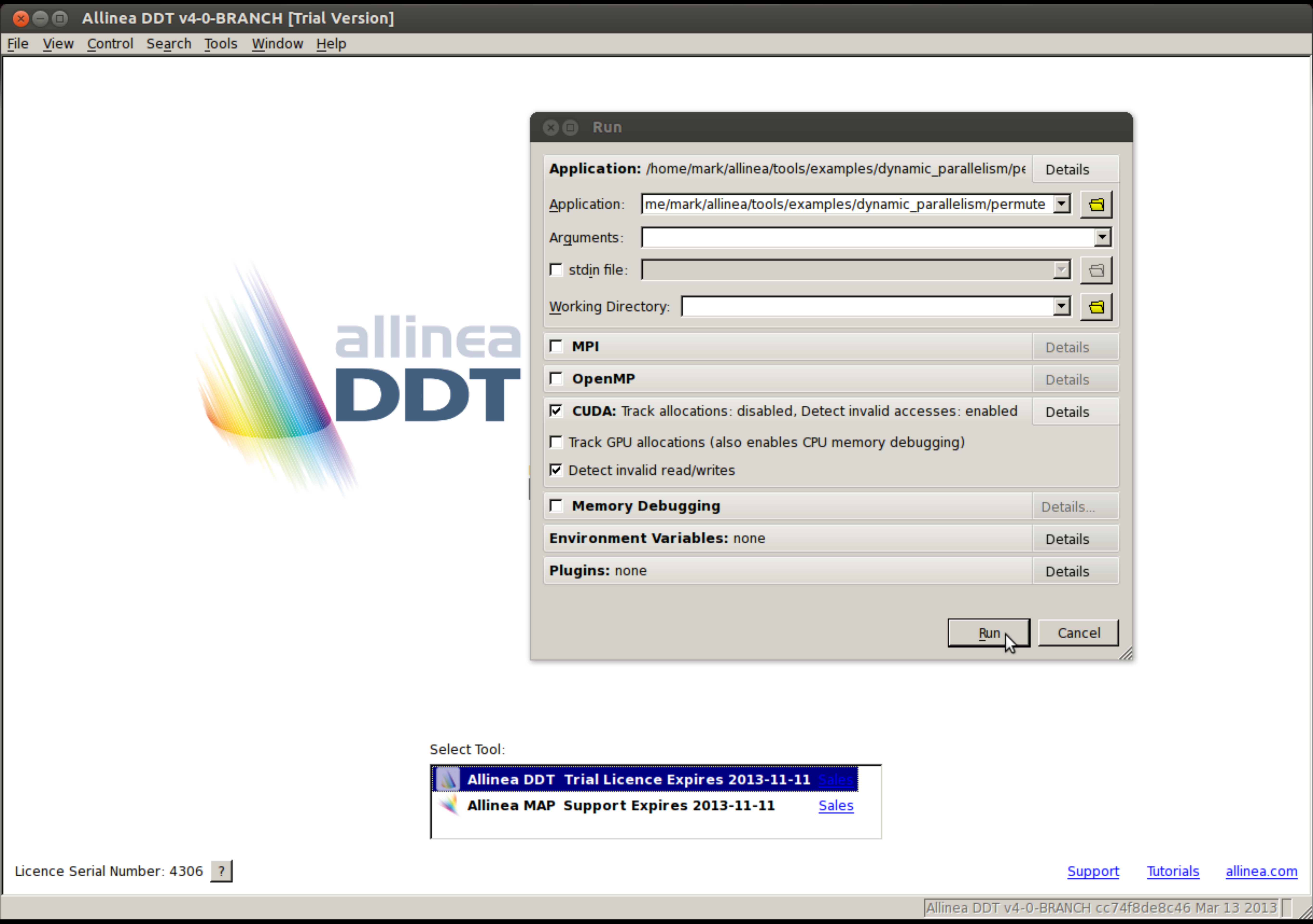
# Debugging is About Understanding



```
smem[threadIdx.x] *= 2; // permute data in
__syncthreads();
// Write back to GMEM since we can't pass
data[threadIdx.x] = smem[threadIdx.x];
__syncthreads();
if (threadIdx.x == 0) {
    permute<<< 1, 256 >>>(n/2, data);
    permute<<< 1, 256 >>>(n/2, data+n/2);
    cudaDeviceSynchronize();
}
```

What actually happens here?

- Which values are put into `data` and when?
- What's the relationship between `n` and `data`?
- How many kernels are launched?



Allinea DDT v4-0-BRANCH [Trial Version]

Run

**Application:** /home/mark/allinea/tools/examples/dynamic_parallelism/pe

Application: me/mark/allinea/tools/examples/dynamic_parallelism/permute

Arguments:

□ stdin file:

Working Directory:

□ **MPI**

□ **OpenMP**

☑ **CUDA:** Track allocations: disabled, Detect invalid accesses: enabled

□ Track GPU allocations (also enables CPU memory debugging)

☑ Detect invalid read/writes

□ **Memory Debugging**

**Environment Variables:** none

**Plugins:** none

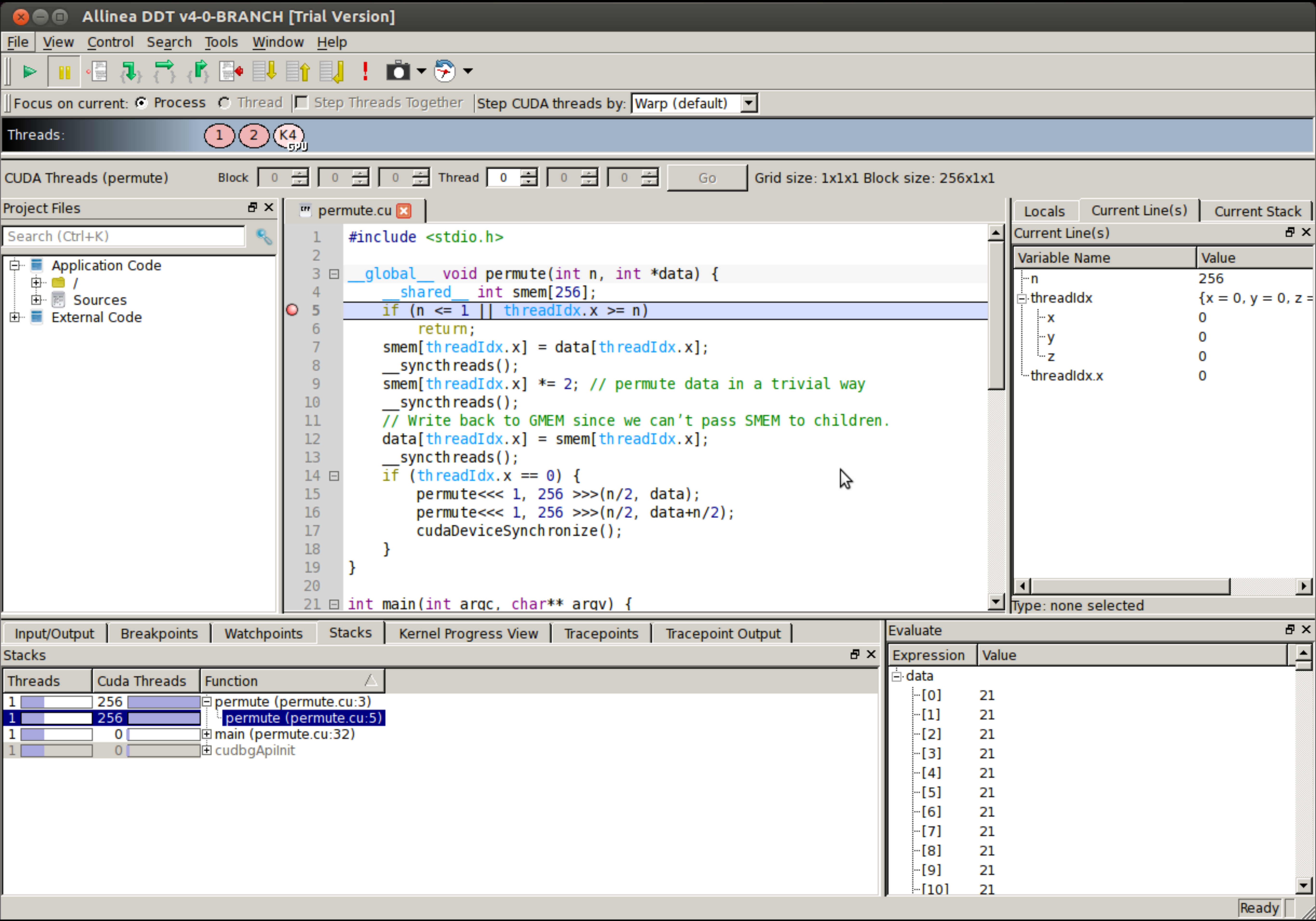
Select Tool:

**Allinea DDT Trial Licence Expires 2013-11-11** Sales

**Allinea MAP Support Expires 2013-11-11** Sales

Licence Serial Number: 4306

Allinea DDT v4-0-BRANCH cc74f8de8c46 Mar 13 2013



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 0 0 0 Grid size: 1x1x1 Block size: 256x1x1

```
#include <stdio.h>

__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1 || threadIdx.x >= n)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}

int main(int argc, char** argv) {
```

Current Line(s)

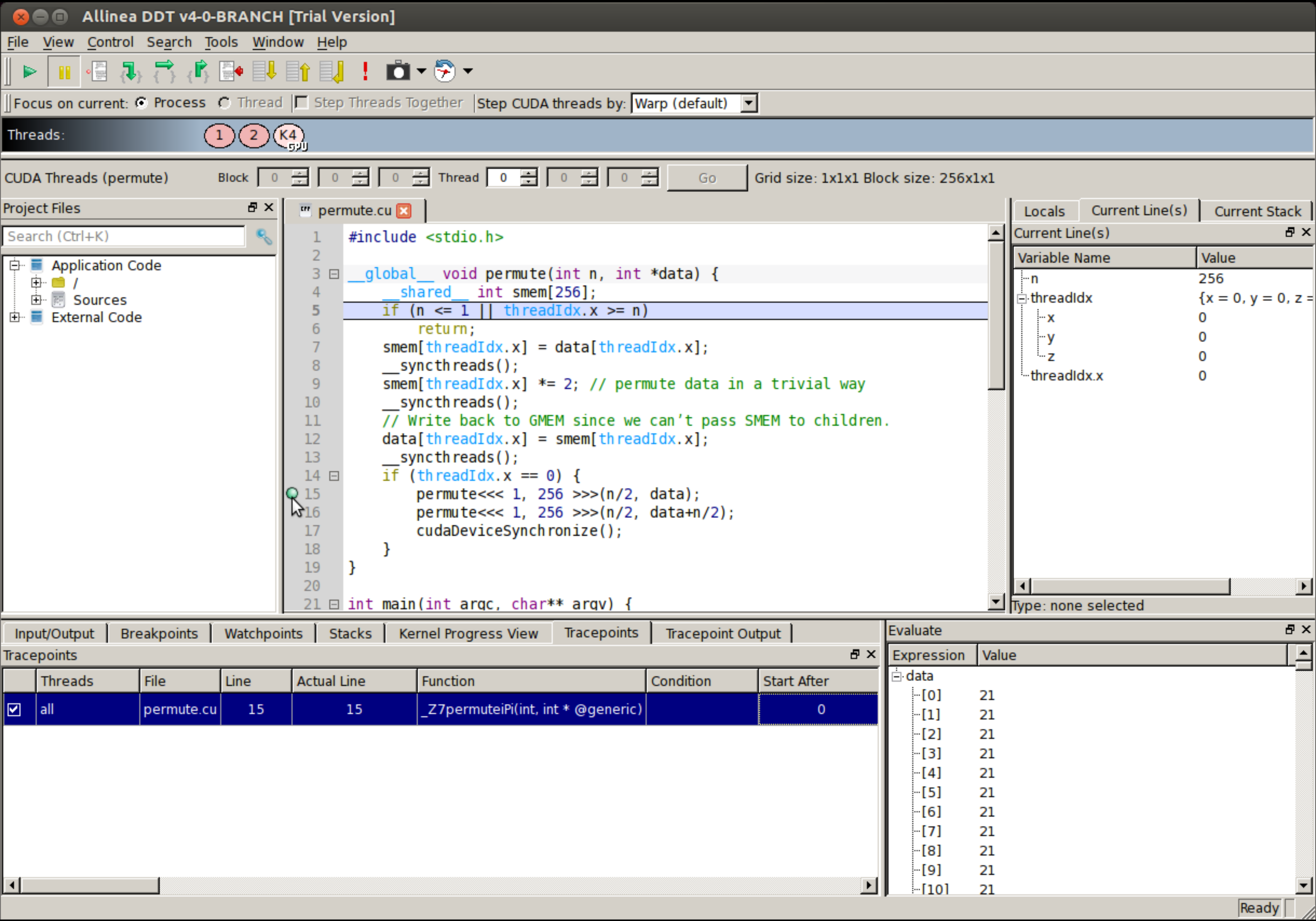
| Variable Name | Value |
|---|---|
| n | 256 |
| threadIdx | {x = 0, y = 0, z = |
| x | 0 |
| y | 0 |
| z | 0 |
| threadIdx.x | 0 |

Type: none selected

Stacks

| Threads | Cuda Threads | Function |
|---|---|---|
| 1 | 256 | permute (permute.cu:3) |
| 1 | 256 | permute (permute.cu:5) |
| 1 | 0 | main (permute.cu:32) |
| 1 | 0 | cudbgApiInit |

| Expression | Value |
|---|---|
| data | |
| [0] | 21 |
| [1] | 21 |
| [2] | 21 |
| [3] | 21 |
| [4] | 21 |
| [5] | 21 |
| [6] | 21 |
| [7] | 21 |
| [8] | 21 |
| [9] | 21 |
| [10] | 21 |



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 0 0 0 Grid size: 1x1x1 Block size: 256x1x1

```
#include <stdio.h>

__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1 || threadIdx.x >= n)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}

int main(int argc, char** argv) {
```

Current Line(s)

| Variable Name | Value |
|---|---|
| n | 256 |
| threadIdx | {x = 0, y = 0, z = |
| x | 0 |
| y | 0 |
| z | 0 |
| threadIdx.x | 0 |

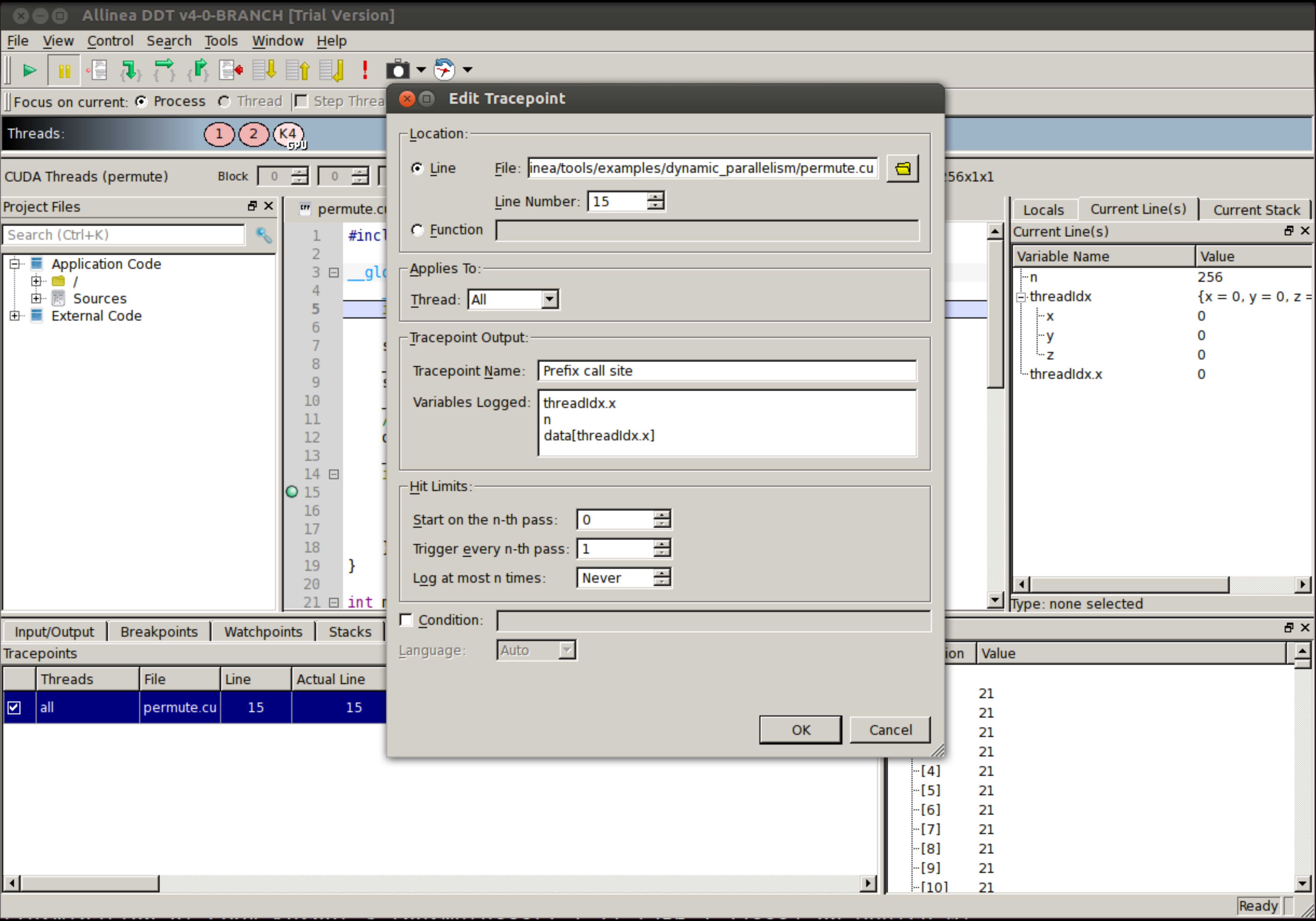
Type: none selected

Tracepoints

| | Threads | File | Line | Actual Line | Function | Condition | Start After |
|---|---|---|---|---|---|---|---|
| ☑ | all | permute.cu | 15 | 15 | _Z7permuteiPi(int, int * @generic) | | 0 |

Evaluate

| Expression | Value |
|---|---|
| data | |
| [0] | 21 |
| [1] | 21 |
| [2] | 21 |
| [3] | 21 |
| [4] | 21 |
| [5] | 21 |
| [6] | 21 |
| [7] | 21 |
| [8] | 21 |
| [9] | 21 |
| [10] | 21 |


Allinea DDT v4-0-BRANCH [Trial Version]
File View Control Search Tools Window Help
Focus on current: ● Process ○ Thread □ Step Threads
Threads: 1 2 K4 GPU
CUDA Threads (permute) Block 0 0
256x1x1
Project Files
Search (Ctrl+K)
Application Code
/
Sources
External Code
permute.cu
Locals Current Line(s) Current Stack
Input/Output Breakpoints Watchpoints Stacks
Type: none selected
Ready
OK
Cancel

Edit Tracepoint

Location:

● Line File: inea/tools/examples/dynamic_parallelism/permute.cu

Line Number: 15

○ Function

Applies To:

Thread: All

Tracepoint Output:

Tracepoint Name: Prefix call site

Variables Logged:
threadIdx.x
n
data[threadIdx.x]

Hit Limits:

Start on the n-th pass: 0

Trigger every n-th pass: 1

Log at most n times: Never

□ Condition:

Language: Auto

Current Line(s)

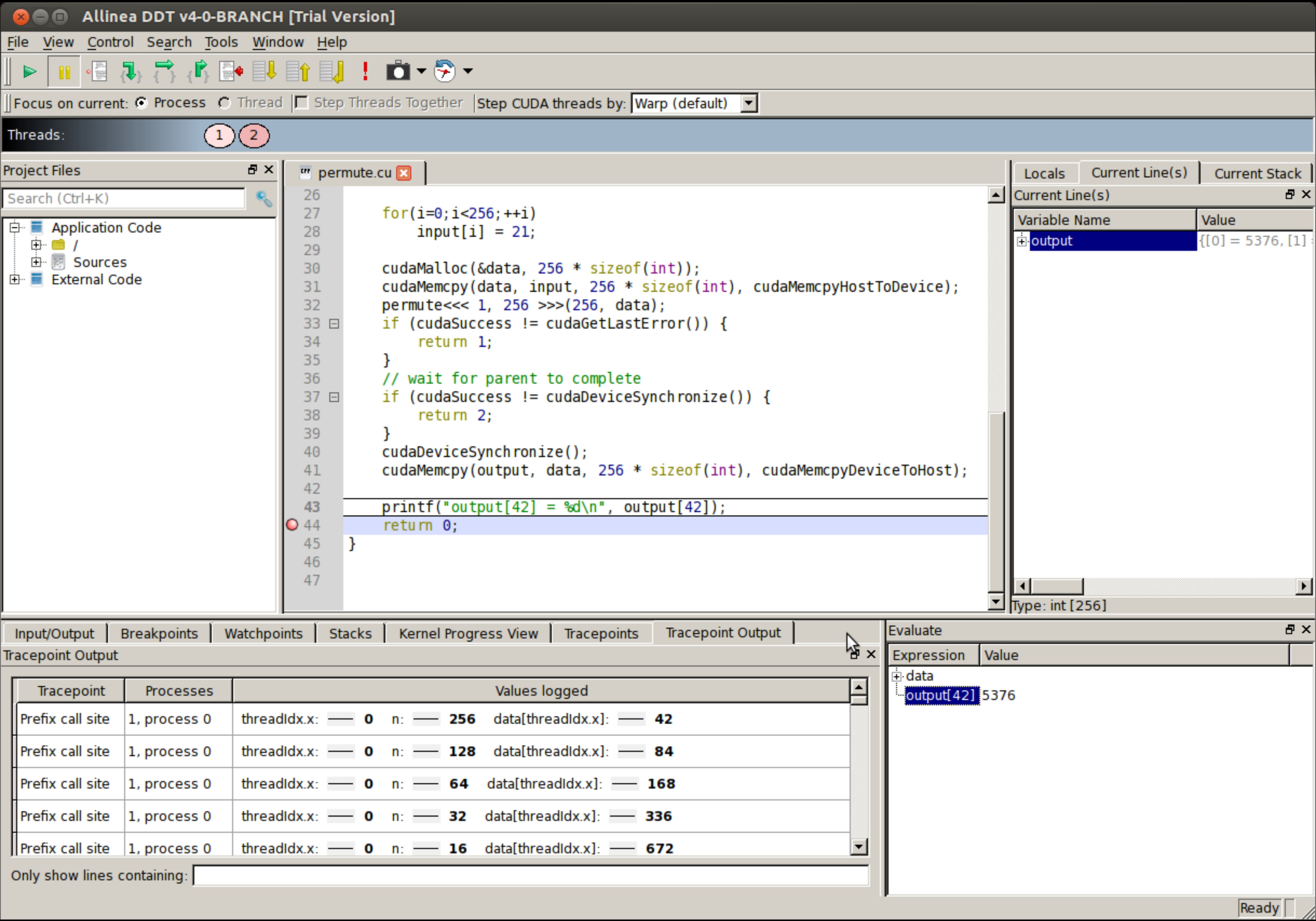
| Variable Name | Value |
|---|---|
| n | 256 |
| threadIdx | {x = 0, y = 0, z = |
| x | 0 |
| y | 0 |
| z | 0 |
| threadIdx.x | 0 |

Tracepoints

|  | Threads | File | Line | Actual Line |
|---|---|---|---|---|
| ☑ | all | permute.cu | 15 | 15 |

| Expression | Value |
|---|---|
|  | 21 |
|  | 21 |
|  | 21 |
|  | 21 |
| [4] | 21 |
| [5] | 21 |
| [6] | 21 |
| [7] | 21 |
| [8] | 21 |
| [9] | 21 |
| [10] | 21 |

```
#incl
__glo
int r
}
```


Allinea DDT v4-0-BRANCH [Trial Version]
File View Control Search Tools Window Help
Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)
Threads: 1 2
Project Files
Search (Ctrl+K)
Application Code
/
Sources
External Code
permute.cu
Locals Current Line(s) Current Stack
Current Line(s)
Input/Output Breakpoints Watchpoints Stacks Kernel Progress View Tracepoints Tracepoint Output
Tracepoint Output
Evaluate
Only show lines containing:
Ready

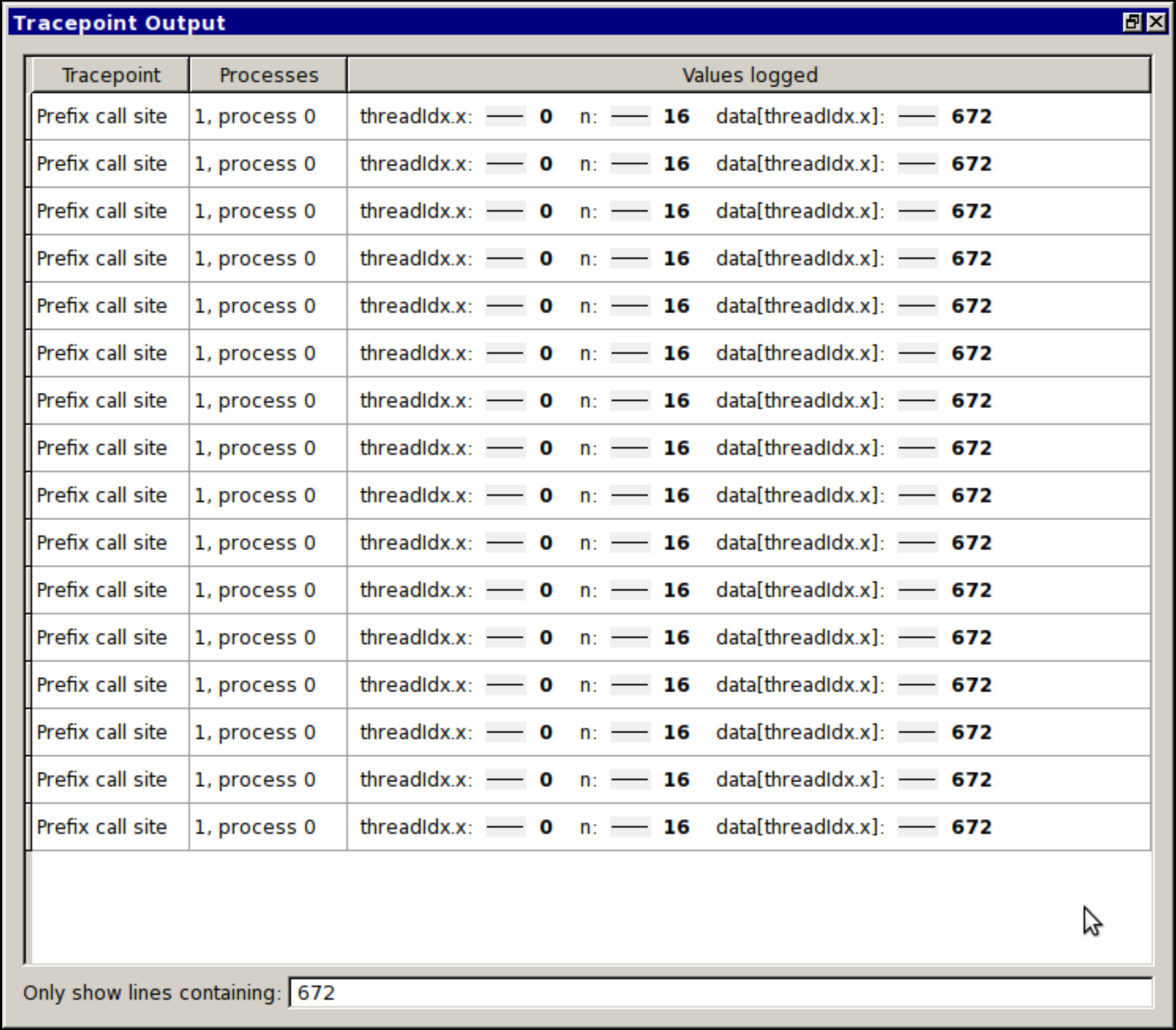
```
    for(i=0;i<256;++i)
        input[i] = 21;

    cudaMalloc(&data, 256 * sizeof(int));
    cudaMemcpy(data, input, 256 * sizeof(int), cudaMemcpyHostToDevice);
    permute<<< 1, 256 >>>(256, data);
    if (cudaSuccess != cudaGetLastError()) {
        return 1;
    }
    // wait for parent to complete
    if (cudaSuccess != cudaDeviceSynchronize()) {
        return 2;
    }
    cudaDeviceSynchronize();
    cudaMemcpy(output, data, 256 * sizeof(int), cudaMemcpyDeviceToHost);

    printf("output[42] = %d\n", output[42]);
    return 0;
}
```

| Variable Name | Value |
|---|---|
| output | {[0] = 5376, [1] |

Type: int [256]

| Tracepoint | Processes | Values logged |
|---|---|---|
| Prefix call site | 1, process 0 | threadIdx.x: 0 n: 256 data[threadIdx.x]: 42 |
| Prefix call site | 1, process 0 | threadIdx.x: 0 n: 128 data[threadIdx.x]: 84 |
| Prefix call site | 1, process 0 | threadIdx.x: 0 n: 64 data[threadIdx.x]: 168 |
| Prefix call site | 1, process 0 | threadIdx.x: 0 n: 32 data[threadIdx.x]: 336 |
| Prefix call site | 1, process 0 | threadIdx.x: 0 n: 16 data[threadIdx.x]: 672 |

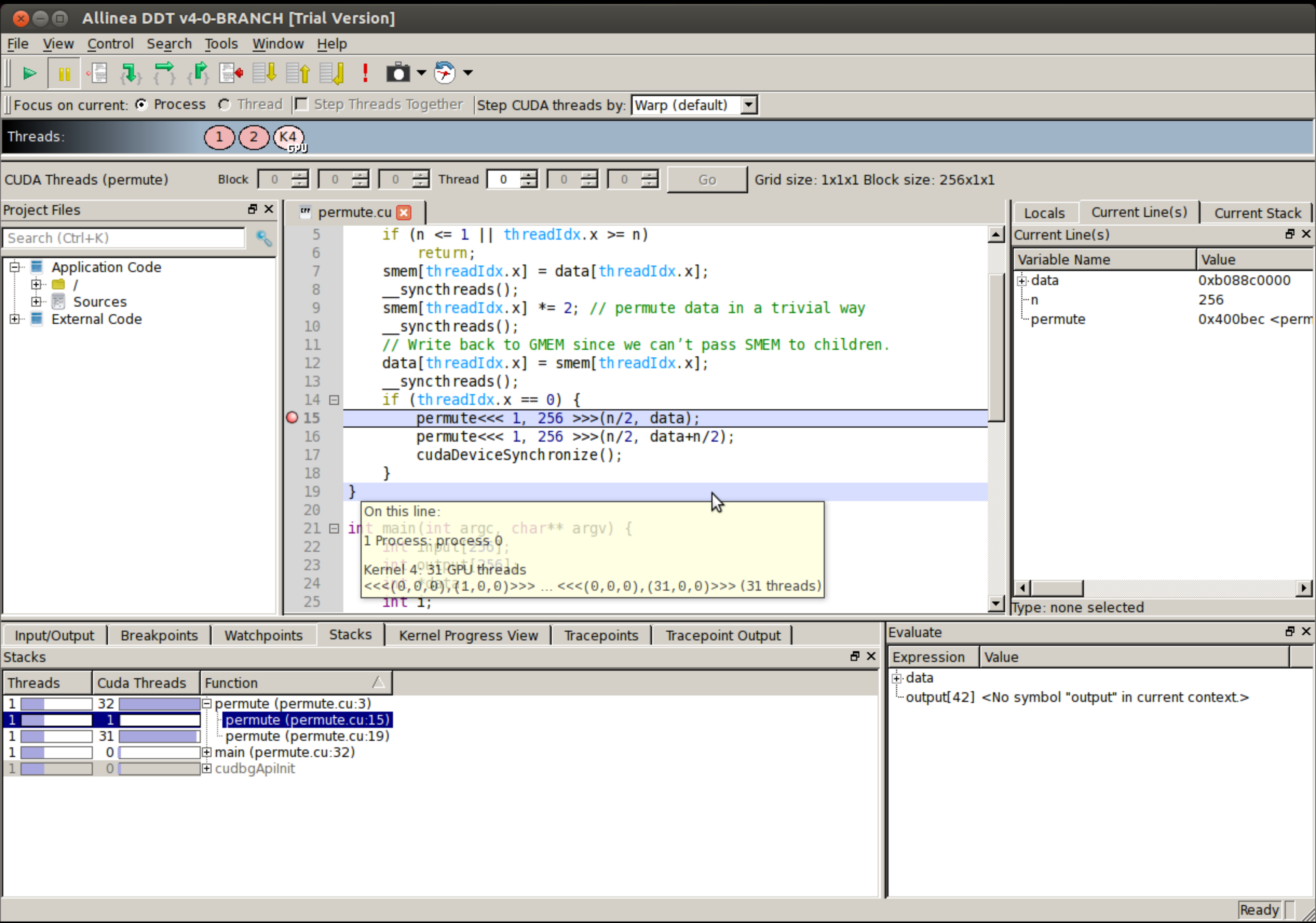
| Expression | Value |
|---|---|
| data | |
| output[42] | 5376 |



Tracepoint Output

| Tracepoint | Processes | Values logged |
|---|---|---|
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |
| Prefix call site | 1, process 0 | threadIdx.x: — 0 n: — 16 data[threadIdx.x]: — 672 |

Only show lines containing: 672



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 0 0 0 Grid size: 1x1x1 Block size: 256x1x1

```
    if (n <= 1 || threadIdx.x >= n)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}

int main(int argc, char** argv) {
    int input[256];
    int output[256];
    int *data;
    int i;
```

On this line:

1 Process: process 0

Kernel 4: 31 GPU threads
<<<(0,0,0),(1,0,0)>>> ... <<<(0,0,0),(31,0,0)>>> (31 threads)

Current Line(s)

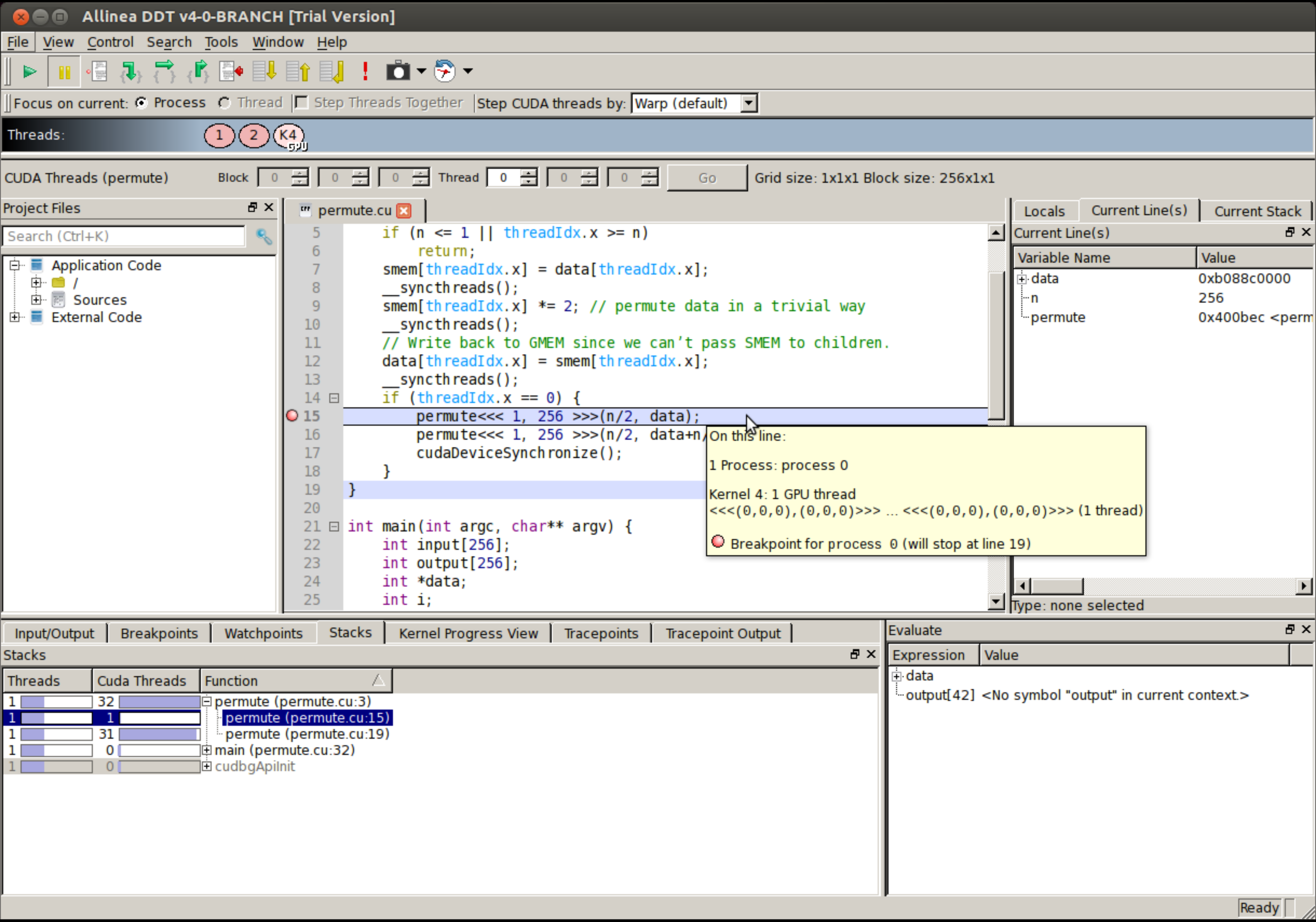
| Variable Name | Value |
|---|---|
| data | 0xb088c0000 |
| n | 256 |
| permute | 0x400bec <perm |

Type: none selected

Stacks

| Threads | Cuda Threads | Function |
|---|---|---|
| 1 | 32 | permute (permute.cu:3) |
| 1 | 1 | permute (permute.cu:15) |
| 1 | 31 | permute (permute.cu:19) |
| 1 | 0 | main (permute.cu:32) |
| 1 | 0 | cudbgApiInit |

| Expression | Value |
|---|---|
| data | |
| output[42] | <No symbol "output" in current context.> |



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 0 0 0 Go Grid size: 1x1x1 Block size: 256x1x1

```
 5      if (n <= 1 || threadIdx.x >= n)
 6          return;
 7      smem[threadIdx.x] = data[threadIdx.x];
 8      __syncthreads();
 9      smem[threadIdx.x] *= 2; // permute data in a trivial way
10      __syncthreads();
11      // Write back to GMEM since we can't pass SMEM to children.
12      data[threadIdx.x] = smem[threadIdx.x];
13      __syncthreads();
14      if (threadIdx.x == 0) {
15          permute<<< 1, 256 >>>(n/2, data);
16          permute<<< 1, 256 >>>(n/2, data+n/
17          cudaDeviceSynchronize();
18      }
19  }
20
21  int main(int argc, char** argv) {
22      int input[256];
23      int output[256];
24      int *data;
25      int i;
```

On this line:

1 Process: process 0

Kernel 4: 1 GPU thread
<<<(0,0,0),(0,0,0)>>> ... <<<(0,0,0),(0,0,0)>>> (1 thread)

● Breakpoint for process 0 (will stop at line 19)

Current Line(s)

| Variable Name | Value |
|---|---|
| data | 0xb088c0000 |
| n | 256 |
| permute | 0x400bec <perm |

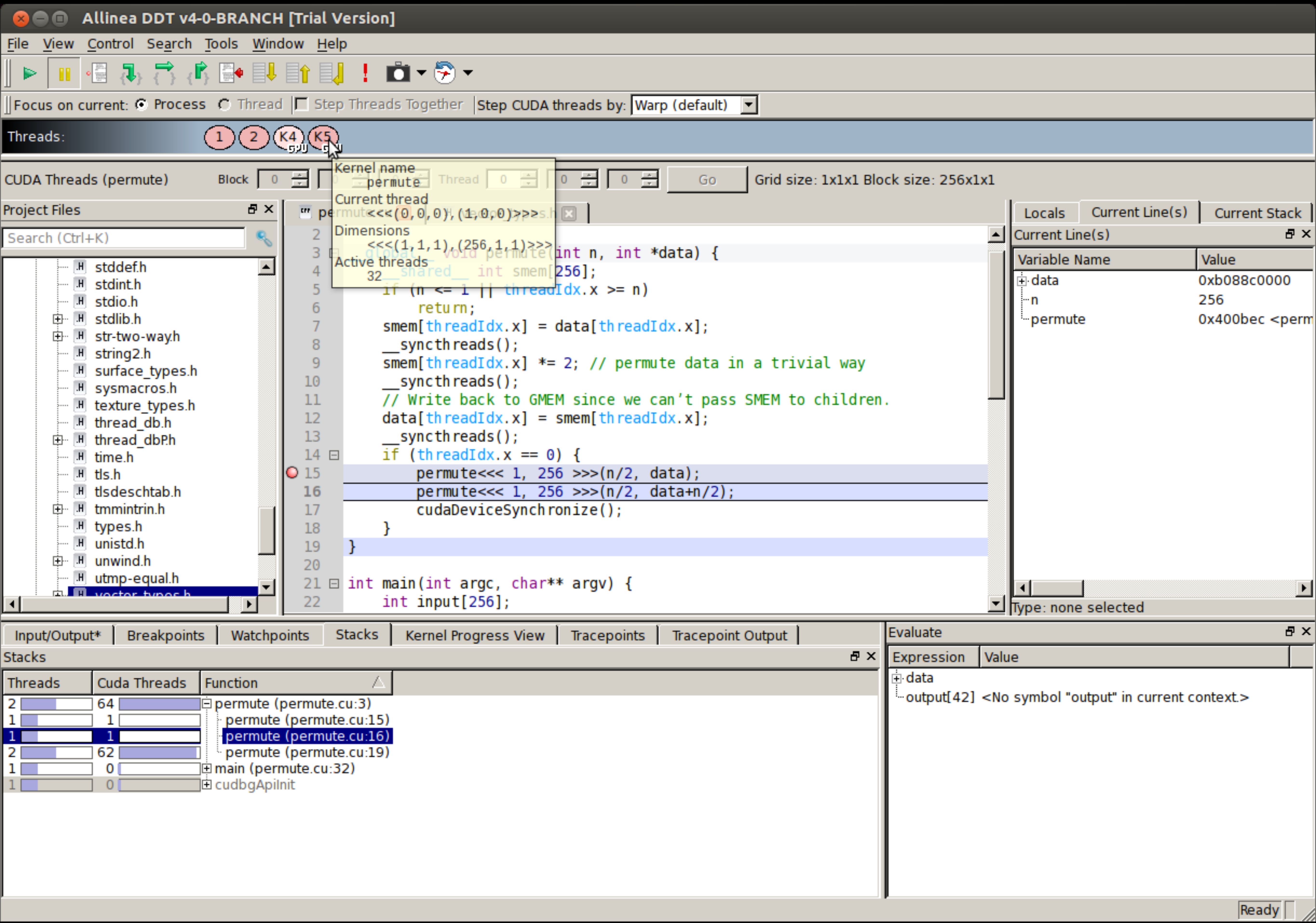
Type: none selected

Stacks

| Threads | Cuda Threads | Function |
|---|---|---|
| 1 | 32 | permute (permute.cu:3) |
| 1 | 1 | permute (permute.cu:15) |
| 1 | 31 | permute (permute.cu:19) |
| 1 | 0 | main (permute.cu:32) |
| 1 | 0 | cudbgApiInit |

Evaluate

| Expression | Value |
|---|---|
| data | |
| output[42] | <No symbol "output" in current context.> |



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 Thread 0 0 0 Grid size: 1x1x1 Block size: 256x1x1

Kernel name
permute
Current thread
<<<(0,0,0),(1,0,0)>>>
Dimensions
<<<(1,1,1),(256,1,1)>>>
Active threads
32

```
__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1 || threadIdx.x >= n)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}

int main(int argc, char** argv) {
    int input[256];
```

Current Line(s)

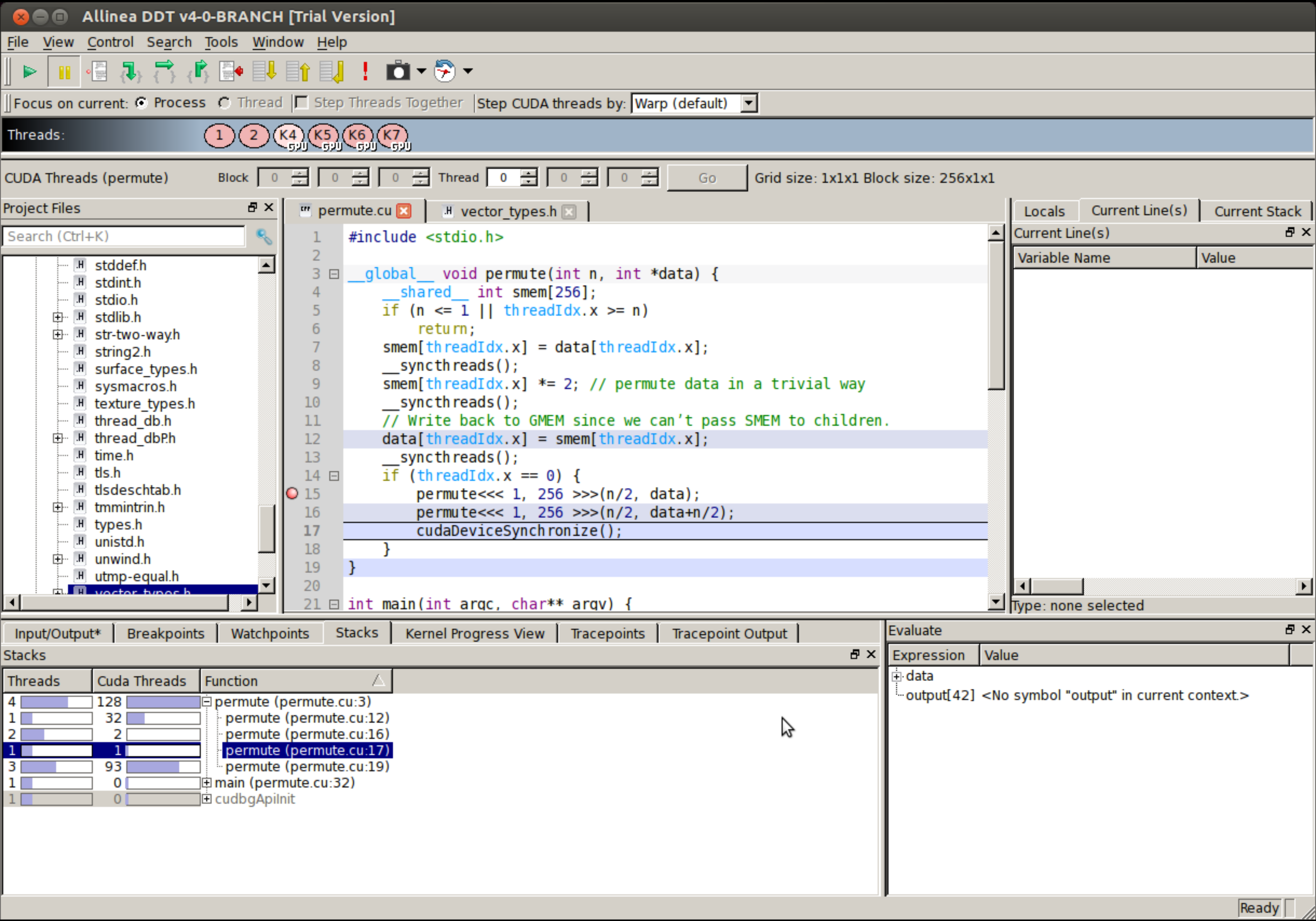
| Variable Name | Value |
|---|---|
| data | 0xb088c0000 |
| n | 256 |
| permute | 0x400bec <perm |

Type: none selected

Stacks

| Threads | Cuda Threads | Function |
|---|---|---|
| 2 | 64 | permute (permute.cu:3) |
| 1 | 1 | permute (permute.cu:15) |
| 1 | 1 | permute (permute.cu:16) |
| 2 | 62 | permute (permute.cu:19) |
| 1 | 0 | main (permute.cu:32) |
| 1 | 0 | cudbgApiInit |

| Expression | Value |
|---|---|
| data | |
| output[42] | <No symbol "output" in current context.> |



Focus on current: ● Process ○ Thread □ Step Threads Together Step CUDA threads by: Warp (default)

CUDA Threads (permute) Block 0 0 0 Thread 0 0 0 Go Grid size: 1x1x1 Block size: 256x1x1

```
#include <stdio.h>

__global__ void permute(int n, int *data) {
    __shared__ int smem[256];
    if (n <= 1 || threadIdx.x >= n)
        return;
    smem[threadIdx.x] = data[threadIdx.x];
    __syncthreads();
    smem[threadIdx.x] *= 2; // permute data in a trivial way
    __syncthreads();
    // Write back to GMEM since we can't pass SMEM to children.
    data[threadIdx.x] = smem[threadIdx.x];
    __syncthreads();
    if (threadIdx.x == 0) {
        permute<<< 1, 256 >>>(n/2, data);
        permute<<< 1, 256 >>>(n/2, data+n/2);
        cudaDeviceSynchronize();
    }
}

int main(int argc, char** argv) {
```

Type: none selected

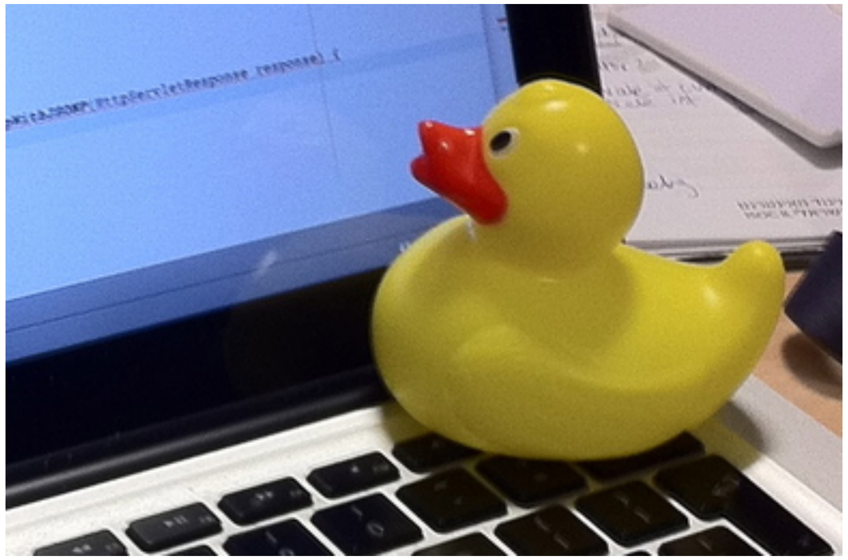
Stacks

| Threads | Cuda Threads | Function |
|---|---|---|
| 4 | 128 | permute (permute.cu:3) |
| 1 | 32 | permute (permute.cu:12) |
| 2 | 2 | permute (permute.cu:16) |
| 1 | 1 | permute (permute.cu:17) |
| 3 | 93 | permute (permute.cu:19) |
| 1 | 0 | main (permute.cu:32) |
| 1 | 0 | cudbgApiInit |

| Expression | Value |
|---|---|
| data | |
| output[42] | <No symbol "output" in current context.> |

# Alternative sources of inspiration

1. Explain the problem to a **rubber duck**.



2. Search your **logbooks**:

```
$ grep -ir blas4a logs/*
segfault-at-4096-procs: Hypothesis: Crash is in blas4a/blas4b
segfault-at-4096-procs: Observation: rank 9 at blas4a.c:145
segfault-at-4096-procs: Fix: Switched to ISend in blas4a and
deadlock-with-openmpi: so the send buffers in blas4a overran.
```

# Summary: Inspiration

When you have a sense for what the problem is:

Test it:
```
$ ddt -offline log.html -trace-at mmult.c:412,rx,ry,rz
```

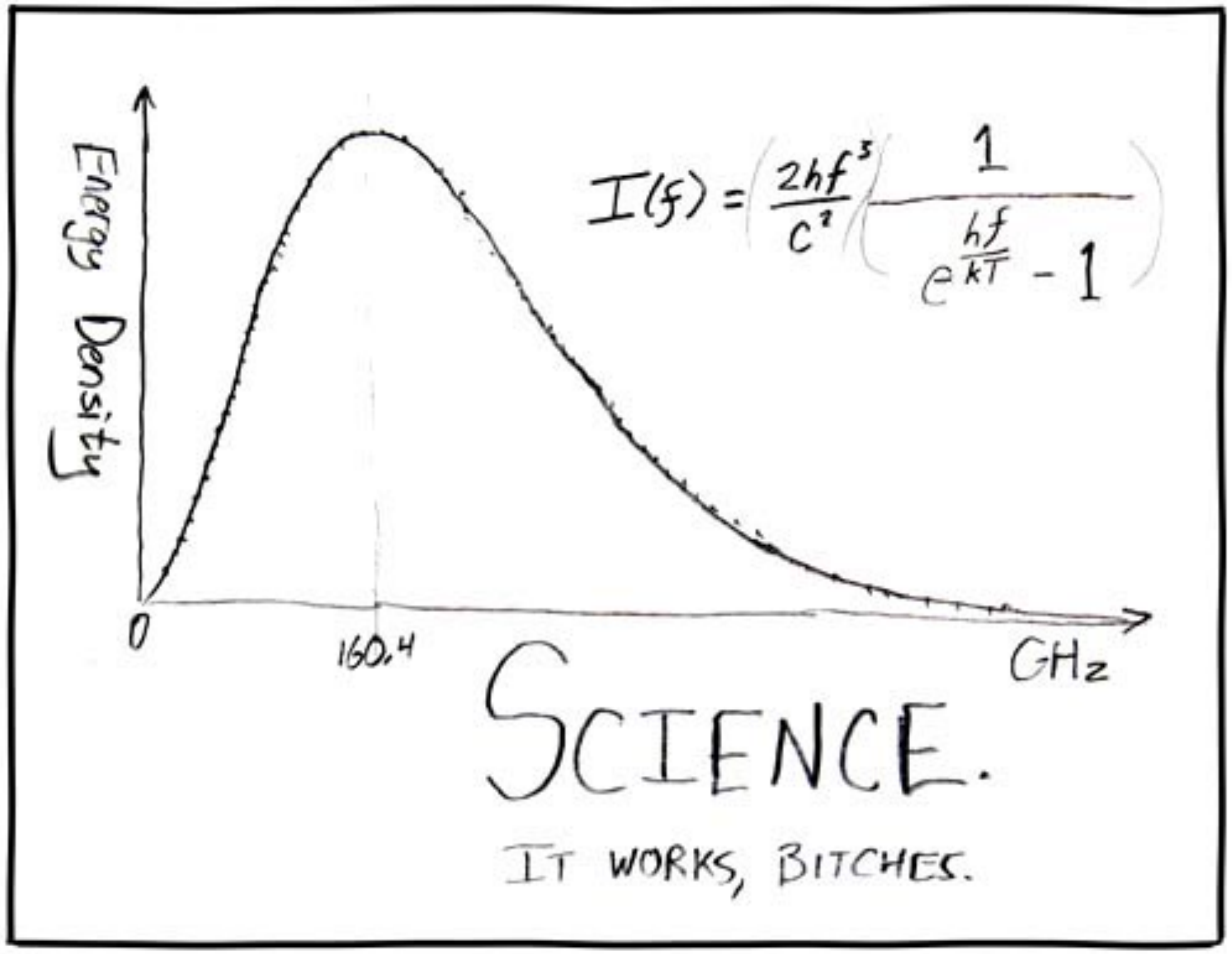
Log it:
```
$ cat >> logs/short-problem-name
    Suspect rx is out of bounds in mmult.c:412.
    Testing with -trace-at mmult.c:412,rx,ry,rz showed...
```

Search your logbooks:
```
$ grep -ri "out of bounds" logs/*
```

Talk to a rubber duck.

**Tip** - set a **time limit** for debugging by inspiration. After 15 minutes, try science.

# Debugging by Science


Energy Density
$I(f) = \left(\frac{2hf^3}{c^2}\right)\left(\frac{1}{e^{\frac{hf}{kT}} - 1}\right)$
0
160.4
GHz
SCIENCE.
IT WORKS, BITCHES.

1. Hypothesis
2. Prediction
3. Experiment
4. Observation
5. Conclusion

There is a **reason** for the bug and you **will** find it!

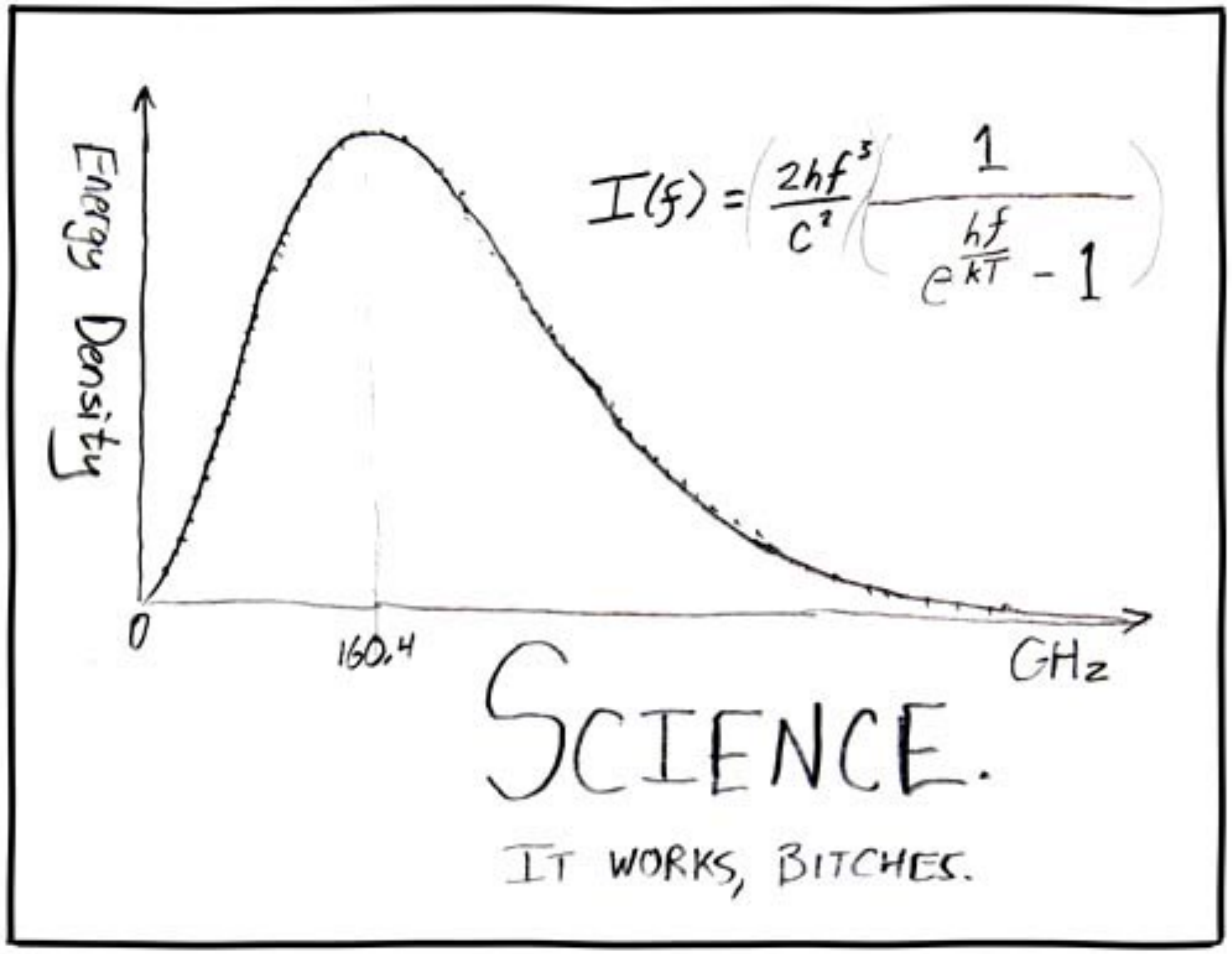
# Science: Your logbook

A logbook is at the heart of debugging by science:

```
hypothesis: cause is in shell_sort()
prediction: At sort.c:6, expect a[] = [11, 4] and size = 2
experiment: -trace-at sort.c:6,a[0],a[1],size
observation: a[] = [11, 14, ?] and size = 3
conclusion: rejected

hypothesis: calling shell_sort with size=3 causes failure
prediction: setting size=2 should make program work
experiment: Set size=2 before call using debugger
observation: As predicted
conclusion: confirmed
```

STAND BACK
I'M GOING TO TRY
SCIENCE

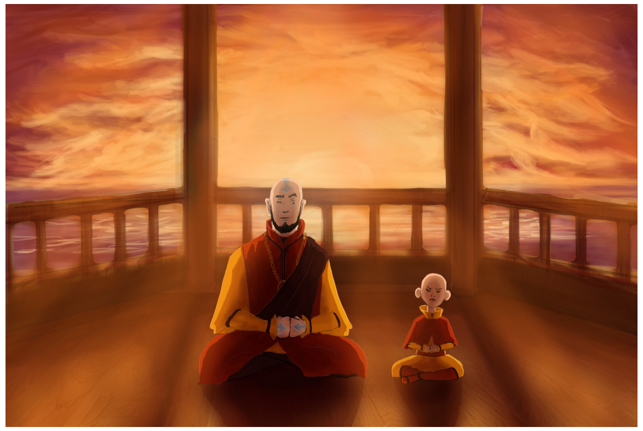
# Summary: Science

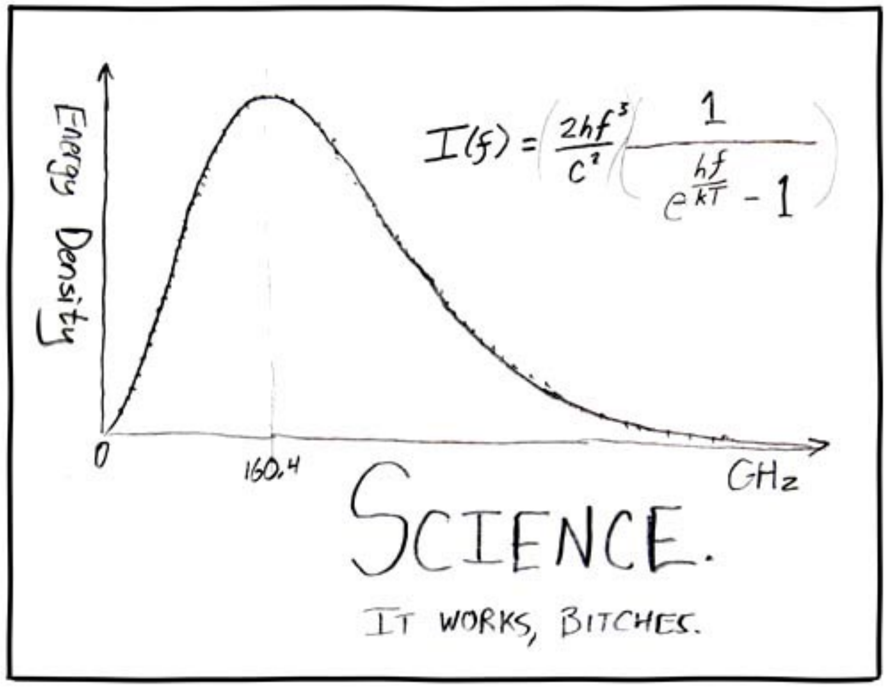

Energy Density
$I(f) = \left(\frac{2hf^3}{c^2}\right)\left(\frac{1}{e^{\frac{hf}{kT}} - 1}\right)$
0
160.4
GHz
SCIENCE.
IT WORKS, BITCHES.

1. Hypothesis
2. Prediction
3. Experiment
4. Observation
5. Conclusion

The method works, but it may not be as direct as you'd like!

# Discipline, Magic, Inspiration, Science




Energy Density
$I(f) = \left(\frac{2hf^3}{c^2}\right)\left(\frac{1}{e^{\frac{hf}{kT}} - 1}\right)$
0
160.4
GHz
SCIENCE.
IT WORKS, BITCHES.

# Thank-you! Questions?

Discipline, Magic, Inspiration and Science

Mark O'Connor
mark@allinea.com